Kinetic approaches for the simulation of non-linear free surface flow problems in civil and environmental engineering

Von der
Fakultät Architektur, Bauingenieurwesen und Umweltwissenschaften
der Technischen Universität Carolo-Wilhelmina
zu Braunschweig

zur Erlangung des Grades eines
Doktoringenieurs (Dr.-Ing.)
genehmigte

Dissertation

von
Christian F. Janßen
geboren am 30.01.1983
in Helmstedt

Eingereicht am	05. November 2010
Disputation am	23. November 2010
Berichterstatter	Prof. Dr.-Ing. habil. Manfred Krafczyk
	Prof. Dr. Stéphan T. Grilli

2011

That's what's nice about writing a thesis and having it done:

There's nothing more I can do about it.

It's done. That's it.

All I can do is let it go
and hope that people are kind to it.

Freely adopted from Tom Hanks.

Abstract

This thesis focuses on the numerical simulation of non-linear free surface flow problems. Different simulation kernels based on the Lattice Boltzmann method (LBM) have been developed or extended, implemented, and, after validation, applied to a number of applications in civil and environmental engineering. The LB model solves viscous and turbulent flows, essentially representing similar physics as Navier-Stokes or reduced shallow water models, but with specific solver advantages concerning data locality and parallel computing. For the advection of the phase interface at the free surface, a hybrid model is developed in this work.

The first part of this thesis deals with the simulation of flow problems on high-performance GPU (graphics processing unit) hardware. First, validations and applications of a reduced LB model for the GPU-accelerated numerical solution of the shallow water equations are presented. The resulting GPU kernel has shown to be applicable to state-of-the-art benchmark problems defined by the tsunami community, dealing with wave propagation and wave run-up. Second, the GPU implementation of a three-dimensional numerical wave tank for the simulation of turbulent flow past a weir, dam break scenarios, wave impact and other applications in civil engineering is presented. For the wave generation, numerical piston-type wave makers have been implemented. The resulting overall performance of both solvers on GPU hardware is found to be at least one order of magnitude higher than of shared-memory-parallel CPU LB kernels on a modern multi-core architecture. The ratio of the numerical simulation runtimes and the real-world time scale of the actual problem could be reduced to a factor of ten for some of the 3D applications and even below unity for wave propagation problems based on reduced models.

The second main target of this thesis is to develop and apply a novel model based on an enhanced representation and advection of the phase interface for the simulation of more complex and demanding free surface flow problems. A volume-of-fluid (VOF) approach in combination with a piecewise linear interface reconstruction (PLIC) has been coupled with the LBM, which provides the solution of the flow field in terms of velocity and density fields. For the implementation, the CPU-based non-uniform LB framework VirtualFluids is used. The resulting hybrid model has been validated against various benchmark test cases and experimental results, showing its validity and applicability. Even a breaking wave during shoaling on a slope, which is a demanding test case for VOF solvers due to high interface curvature and velocity gradients, was successfully simulated. Apart from the model development and validation itself, a coupling to a rigid body engine for the simulation of FSI problems has been established. Finally, several techniques for the coupling to a potential flow solver are discussed and validated, in order to generate realistic wave profiles and for the efficient simulation of wave run-up and wave breaking.

Kurzfassung

Die vorliegende Dissertation behandelt die numerische Simulation von nichtlinearen Strömungen mit freien Oberflächen, wie beispielsweise brechenden Meereswellen, Tsunamis oder Dammbruchszenarien. Dazu werden verschiedene Simulationskerne auf Basis der Gitter-Boltzmann-Methode (engl. Lattice Boltzmann method, LBM) entwickelt, implementiert und nach ihrer Validierung auf zahlreiche Aufgabenstellungen im Bau- und Umweltingenieurwesen angewendet. Die LB-Methode wird verwendet, um viskose und turbulente Strömungen numerisch zu simulieren und beinhaltet eine physikalische Beschreibung, die gewöhnlich mit Modellen auf Basis der Navier-Stokes- oder reduzierten Flachwassergleichungen umgesetzt wird. Sie bietet dabei löserspezifische Vorteile bezüglich der Datenlokalität und dem parallelen Rechnen. Für die Advektion der freien Oberfläche wird im Rahmen dieser Arbeit ein hybrides Modell entwickelt.

Der erste Teil der Arbeit beschäftigt sich mit der Simulation von Strömungsproblemen auf High-Performance-GPU (Graphics Processing Unit) Hardware. Einleitend wird die Validierung und Anwendung eines LB-Modells für die Lösung der Flachwassergleichungen dargestellt. Der resultierende GPU-Kernel ist auf aktuelle Benchmark-Probleme der Tsunami-Community wie z.B. Wellenausbreitung oder Wellenauflauf anwendbar. Im Anschluss wird eine GPU-Implementierung eines dreidimensionalen numerischen Wellenkanals für die Simulation turbulenter Wehrströmungen, Dammbruchszenarien, des Wellenschlages auf Pfahlbauwerke und anderer Anwendungen im Bauingenieurwesen präsentiert. Die resultierende Gesamtrechenleistung der beiden Strömungslöser auf der GPU Hardware ist mindestens um eine Größenordnung höher als bei vergleichbaren CPU-Implementierungen auf einer modernen Mehrkern-Architektur. Der Unterschied zwischen der Simulationsdauer und der Zeitskala der Zielanwendung kann für einige der 3D Anwendungen auf lediglich eine Größenordnung reduziert werden, während die Berechnung der Wellenausbreitungs-Phänomene auf Basis reduzierter Modelle teilweise sogar schneller als in Echtzeit möglich ist.

Das zweite Ziel dieser Arbeit ist die Entwicklung und Anwendung eines neuartigen Modells für die Simulation von komplexen und anspruchsvollen Problemen mit freier Oberfläche unter Zuhilfenahme einer erweiterten Repräsentation der Phasengrenzfläche. Ein Volume-of-Fluid (VOF) Ansatz auf der Grundlage einer abschnittsweise linearen Interface-Rekonstruktion (engl. piecewise linear interface reconstruction, PLIC) wird an die LBM gekoppelt, die weiterhin die Lösung des Strömungsfeldes in Bezug auf Geschwindigkeit und Dichte bereitstellt. Das resultierende hybride Modell wird anhand verschiedener Benchmark-Tests und experimenteller Ergebnisse validiert, und stellt seine Gültigkeit und Anwendbarkeit unter Beweis. Selbst brechende Wellen, die für VOF-Löser aufgrund der hohen Krümmung und Geschwindigkeitsgradienten einen komplexen Testfall darstellen, können erfolgreich simuliert worden. Im Anschluss an die Entwicklung und Validierung des eigentlichen Modells wird eine

Kopplung an einen Starrkörper-Löser realisiert, welche die Simulation von Problemstellungen aus dem Bereich die Fluid-Struktur-Interaktion ermöglicht. Abschließend werden verschiedene Techniken zur Kopplung des hybriden Lösers an einen numerischen Wellenkanal auf Basis der Potentialströmungstheorie diskutiert und validiert. Somit ist die Erzeugung realistischer Wellenprofile und die effiziente Simulation von Wellenauflauf sowie Wellenbrechen möglich.

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für rechnergestützte Modellierung im Bauingenieurwesen der Technischen Universität Braunschweig. An erster Stelle danke ich meinem Doktorvater Herrn Prof. Dr. habil. Manfred Krafczyk für die große Unterstützung während dieser Promotion, vor allem für das große Vertrauen in meine Arbeit und die Freiheit, die ich am iRMB genießen durfte.

Bedanken möchte ich mich auch bei Herrn Prof. Stéphan T. Grilli für die Gastfreundlichkeit, mit der er mich während meiner Promotion als Wissenschaftler in Rhode Island willkommen geheißen hat, für die fachliche Unterstützung und für die Bereitschaft, das Zweitgutachten der Arbeit zu übernehmen.

Ich danke weiterhin den fachlichen Freunden und Weggefährten der letzten Jahre, die mich unabhängig von meinem konkreten Promotionsthema unterstützt haben, unter anderem Prof. Dr. Andreas Zilian, nicht nur für die Bereitschaft, die Aufgabe des Prüfers in meiner Prüfungskommission zu übernehmen.

Mein besonderer Dank gilt meinen Kollegen und Freunden am iRMB, die mich während der letzten Jahre auf dem Weg zur Promotion begleitet und unterstützt haben. Es war nicht immer leicht, manchmal hart, aber immer herzlich. Ich danke Dr. Sebastian Geller und Dr. Sören Freudiger für die Entwicklung von VirtualFluids und ihre Unterstützung in der Auseinandersetzung mit objektorientierter Programmierung und parallelem Rechnen. Vielen Dank an Jannis Linxweiler für die zahlreichen Gespräche über die Tücken der GPU-Programmierung und an Maik Stiebler für konstruktive Kritik jeglicher Art und viele fruchtbare Diskussionen über Numerik.

Der wichtigste Dank gebührt jedoch meinen Eltern, dafür, dass sie mich immer bedingungslos unterstützt und motiviert haben, egal wie verzwickt die Lage gerade war.

Zuletzt danke ich Peter Lynn dafür, dass er mein Lieblingshobby erfunden hat, Jim Marshall dafür, dass seine Regler bis elf gehen, Leonardo da Vinci dafür, dass es heutzutage Modellhubschrauber gibt, sowie Diedrich König und Theodor Fetköter für das beste Bier der Welt.

Contents

CHAPTER 1

Introduction

Figure 1.1: Two plunging breakers during a tropical storm, Point Judith, RI, USA

1.1 Motivation and background

For more than two millennia, fluid dynamics has been in the center of interest of mankind, starting with Archimedes' famous discovery of the basic principle of buoyancy around 250BC. Since then, especially during the 18th and 19th centuries, the fundamental equations of fluid dynamics have been progressively derived. In the past century, very advanced challenges, such as turbulence, were tackled and still are part of active research, whereas the general existence of a unique solution to the governing fluid equations has not been proven yet, but is to be considered as a Millennium problem. Compared to the ancient research activities, the widely available computational power has changed the way of conducting research nowadays. Numerical simulations have become an inherent part of fluid dynamics-related research, and the computational power, which is currently available, is far from being sufficient for dealing with problems of the desired complexity.

The topic of fluid dynamics is broadly diversified, and various different flow types can be distinguished, whereas this thesis merely is dedicated to free surface flows. Basically, free-surface flows are two-phase flows where high viscosity ratios and high density ratios between the two phases are present. The flow behavior is dominated by the denser phase and the interface is allowed to move freely. If capillary forces are neglected, the simulation of the denser and more viscous phase is sufficient and the influence of the second (less dense) phase on the flow dynamics can then be represented by appropriate boundary conditions at the interface.

Typical free surface flow problems occur in various fields of civil and naval/ocean engineering. Free surface flows, such as breaking waves or tidal currents, are fascinating, as they give us indications of the enormous amount of power that water can contain, for both good and bad reasons. Recently, natural disasters have attracted public attention to hazardous free surface flow events. Particularly, the 2004 Boxing Day event of sinister memory, has increased the awareness of the potential threat posed by tsunamis. Warning systems are currently being installed, demanding both a good probe system and reliable numerical models to predict tsunami severeness and details of coastal impact. These events can only be described by reduced governing equations, as the huge domain size already is very demanding and high grid resolution is necessary.

Apart from these challenging, even life-threatening, long-term wave propagation applications, wave impact and wave slamming play an important role in the design process in various kinds of engineering sciences. For the design of offshore structures, such as offshore wind turbines, which are possibly one answer to the open questions concerning the future of our energy supply, the wave impact force on monopiles is the main load case during the design process and has been part of extensive research. For naval engineers, the flow patterns around and in the wake of a ship hull are important to know and hard to optimize, as they highly depend on turbulent effects in the boundary layer. From a numerical point of view, all those applications require fully three-dimensional, viscous and turbulent free surface flow simulations.

Even in this limited list of applications, the diversity of free surface flow problems and the wide range of length and time scales involved becomes obvious. Moreover, research on free surface is by nature multidisciplinary. Not only regarding flow theory, but also for the insight in the algorithms and high performance computing that are needed for the development of accurate and efficient free surface flow solvers. The main purpose of this work is to provide a reliable set of numerical tools for the

simulation of free surface related flow problems, on different length and time scales. State-of-the-art high performance computing devices will be used, in combination with an efficient numerical method, to address various problems in civil and environmental engineering.

1.2 Purpose

Before going into details, in the following free surface flow parts, the first two chapters review the basics of fluid mechanics and the proposed numerical method (Part I). Chapter 2 summarizes the governing equations of hydrodynamics and free surface hydraulics, focusing on several widely used model simplifications and their interrelations. In chapter 3, the Lattice Boltzmann method (LBM) is introduced. The LBM is a powerful numerical scheme for discretizing flow equations on the basis of a mesoscopic approach. It has been shown to be an efficient approach for solving a variety of demanding CFD problems, including those in the field of multiphysics. The derivation is starting with the basics of lattice gas and ending up with a Chapman Enskog expansion, showing that the model is able to reproduce solutions of the well known Navier-Stokes equations.

The remainder of this thesis is subdivided into two major parts. Part II deals with the simulation of shallow water flows and three- dimensional free surface flows on high-performance GPGPU hardware (General Purpose Graphics Processing Units). Initially, in chapter 4, the multiphysics extensions of the numerical method are briefly reviewed, before showing the implementation in a GPU framework in chapter 5. Validations and applications of a reduced model for tsunami runup problems are presented. Secondly, a three-dimensional numerical wave tank is presented for the simulation of turbulent flow past a weir, dambreak scenarios, and slamming forces applied on structures.

Part III of this work details the derivation, implementation, and application of an enhanced interface model for the simulation of even more complex free surface flows. Numerically, the free surface represents a moving boundary, which is allowed to move freely (under certain kinematic and dynamic boundary conditions). To obtain accurate results, a more general approach has to be pursued by applying state-of-the-art advection schemes and focusing on developing a hybrid LBM-Volume-Of-Fluid (VOF) model, for the application on non-uniform grids. After a basic review of interface capturing methods, an alternative method is developed, on the basis of a piecewise linear interface reconstruction (PLIC) scheme (chapter 6), and coupled to the Lattice Boltzmann flow solver, resulting in a hybrid scheme which is presented in chapter 7. Besides the model development and validation the coupling to a rigid body engine, for the simulation of FSI problems, has been established (chapter 9). Finally, the coupling to a potential flow solver allows for the generation of realistic wave profiles, the efficient simulation of wave run-up, and wave breaking on slopes (chapter 8).

At the end, the thesis concludes with discussions of future developments and applications of the model.

Part I

Basics

CHAPTER 2

Theoretical background

Free surface flow problems have been of particular interest for scientists and researchers for more than a century. Depending on the length scales and phenomena of interest, various models have been used, each of these focusing on different parts of the underlying physics. For the fully non-linear, three-dimensional simulation of the fluid phase, fluid models on the basis of the full Navier-Stokes equations, including viscous effects and effects of turbulence, are typically used. If wave propagation is of interest, simplified flow models, such as depth-averaged equations, have been widely applied. This chapter gives an overview of flow governing equations.

2.1 A brief review of hydrodynamics

Any macroscopically continuous matter or fluid is in fact composed of single molecules. These molecules move and interact with each other, i.e., collide with other particles and with solid objects. Nevertheless, most macroscopic Computational Fluid Dynamics (CFD) approaches neglect the microscopic composition of matter and consider the fluid to be continuous (*continuum hypothesis*). Instead of molecular properties, macroscopic ones (e.g., pressure, velocity, and possibly also density and temperature) are predicted. These are assigned to a continuous field and hence assumed to be continuous. This so-called continuum hypothesis yields very accurate solutions for a wide range of applications. Problems for which the continuum hypothesis does not yield sufficiently accurate solutions are solved using statistical mechanics. In this work, a mesoscopic method is used, on the basis of kinetic gas theory. The latter is presented in detail in chapter 3. In the following, the well-known and widely used governing equations of fluid mechanics are derived on a macroscopic scale.

2.1.1 Navier-Stokes equations and its simplifications

The main basic laws of physics and engineering are conservation of mass, momentum and energy. In isothermal and incompressible CFD simulations, the latter is not considered. In combination with the

conservation laws, the fundamental *Reynolds transport theorem* is the starting point of the derivation in this work.

The *Reynolds transport theorem* is a basic concept of physics for balancing extensive properties Ξ. These properties depend on the size of the system under consideration, such as its mass and volume. If a complete system is doubled, mass and volume is doubled as well. By contrast, intensive properties, noted ξ, do not depend on the system size; these are for instance density, temperature and pressure. If a system with pressure p_0 is duplicated and recombined, the pressure p_0 remains constant. Extensive properties may be transferred to intensive ones via dividing by the system mass:

$$\Xi = \int_{\Omega_c} \rho \cdot \xi \, d\Omega \tag{2.1}$$

For an extensive property Ξ with corresponding intensive property ξ, the Reynolds transport theorem states

$$\frac{D\Xi}{Dt} = \int_{\Omega_c} \partial_t \left(\rho\xi\right) \, d\Omega + \int_{\Gamma_c} \rho\xi \cdot \mathbf{v_b} \cdot \mathbf{n} \, d\Gamma = \int_{\Omega_c} q \, d\Omega \tag{2.2}$$

for the control volume Ω_c with surface Γ_c and a substance of density ρ. $\mathbf{v_b}$ denotes the relative velocity between substance and control volume at the boundary Γ_c, with outward pointing normal vector $\mathbf{n}$. Applying Gauss theorem yields

$$\frac{D\Xi}{Dt} = \int_{\Omega_c} \left\{\partial_t \left(\rho\xi\right) + \nabla \left(\rho\xi \cdot \mathbf{v_b}\right)\right\} d\Omega = \int_{\Omega_c} q \, d\Omega \tag{2.3}$$

This theorem can now be applied to the following physical conservation laws. The resulting set of equations for mass conservation ($\Xi = 1$) and momentum conservation ($\Xi = u$) reads

$$\nabla \mathbf{v} = 0 \tag{2.4}$$

$$\partial_t \mathbf{v} + (\mathbf{v}\nabla)\mathbf{v} = -\frac{1}{\rho_0}\nabla P + \nu_0 \Delta \mathbf{v} \tag{2.5}$$

for an incompressible fluid with constant density ρ and Newtonian viscous material behavior, with kinematic viscosity ν and Laplace operator $\Delta = \nabla^2$. If viscous effects are neglected, the Navier-Stokes equations reduce to the Euler equations

$$\nabla \mathbf{v} = 0 \tag{2.6}$$

$$\partial_t \mathbf{v} + (\mathbf{v}\nabla)\mathbf{v} = -\frac{1}{\rho_0}\nabla p, \tag{2.7}$$

again given for the incompressible case. It follows that incompressible velocity fields always have to be divergence-free in order to satisfy continuity equation. If conservation of energy has to be explicitly satisfied, the same theorem is applied to energy ($\Xi = e$), yielding a third equation.

1D simplification: Bernoulli equations

For stationary and inviscid fluids in one dimension, or more specifically along any streamline or vortex line in a complex stationary flow, one can show that the Euler equations reduce to a simplified formulation:

$$vA = \text{const.} \tag{2.8}$$

$$\frac{v^2}{2g} + \frac{p}{\rho g} + z = \text{const.} \tag{2.9}$$

These so-called Bernoulli equations (see e.g. [117]) are basically stating that the piezometric height, i.e., the sum of velocity height $\frac{v^2}{2g}$, pressure height $\frac{p}{\rho g}$ and geodetic height z, is constant, with fluid velocity v, gravity g, pressure p, density ρ and geodetic level z. The velocity height is related to the dynamic pressure, the pressure height is related to the static pressure.

Bernoulli's equation can be derived in several ways, for instance in 1D by integrating Euler equations under the assumption of constant velocities and pressures in y and z direction. During the derivation, a Weber transformation of the momentum equation is used (stating that $(\mathbf{v} \cdot \nabla)\mathbf{v} = \nabla(\frac{v^2}{2}) + \mathbf{v} \times (\nabla \times \mathbf{v})$). Hence, the equations are only valid along streamlines or vortex lines in rotational flow, which are tangential to the velocity field or vorticity fields, respectively. The equations apply everywhere by definition where the velocity field is irrotational, or for irrotational flows in general. Note, for non-stationary irrotational flows, the equations still apply but one additional term is added. Bernoulli's equations often serve to estimate the overall and general flow behavior and they are intensively used in free surface hydraulics and for example in the design of weir-type structures. Moreover they serve in understanding the basic fluid dynamical relationship between pressure and velocity, stating that an acceleration of the fluid always involves a pressure drop. Eventually, energy considerations on the basis of Bernoulli's equations can serve to estimate the energy distribution in a fluid.

2D simplification: Shallow water equations

When the horizontal flow length scale is much larger than the vertical length scale, which is the case for numerous applications in ocean engineering, a depth-integrated version of Navier-Stokes equations can be used [184]. According to continuity equation, the variations of the vertical velocity component of the flow field are small in such a case. Considering momentum equation, this automatically demands that the vertical pressure gradient is nearly hydrostatic and the dynamic pressure contribution is small. Assuming a constant horizontal velocity field over depth, a vertical integration of the momentum equation allows to remove the vertical velocity from the equations. Although the vertical velocity is not present in the shallow water equations (note that this velocity is not necessarily zero), it can be obtained in a post-processing step when applying continuity equation. The resulting shallow water equations (SWE) read:

$$\partial_t h + \nabla\left(h\bar{\mathbf{v}}\right) = 0 \tag{2.10}$$

$$\partial_t\left(h\bar{\mathbf{v}}\right) + \nabla\left(h\bar{\mathbf{v}} \cdot \bar{\mathbf{v}}\right) = -\nabla\left(\frac{1}{2}gh^2\right) + \nu\Delta\left(h\bar{\mathbf{v}}\right) + \mathbf{F} \tag{2.11}$$

with a forcing term $\mathbf{F}$ being defined as

$$\mathbf{F} = -gh\nabla z_b + \frac{\tau_w}{\rho} + \frac{\tau_b}{\rho} + \mathbf{E} \tag{2.12}$$

including the effects of bottom elevation z, bed shear stress τ_b, wind shear stress τ_w and the Coriolis force $\mathbf{E}$.

These equations are widely applicable, mainly in the field of ocean engineering and atmospheric modeling, where one length scale is dominant. If vertical variations shall be taken into account, they are separated from the horizontal ones, resulting in a set of shallow water equations for each horizontal fluid layer (multilayer SWE).

Potential flow theory

In potential flow theory, the underlying assumptions are inviscid and irrotational flow. The velocity field is defined as the gradient of a velocity potential $\Phi(x,y,z,t)$, $\mathbf{v} = \nabla\Phi$. Accordingly, continuity equation becomes Laplace's equation for the potential

$$\nabla\mathbf{v} = \nabla\left(\nabla\Phi\right) = \Delta\Phi = 0. \tag{2.13}$$

The flow field can be uniquely determined from its kinematics. Dynamics can be applied in a post-processing step for the computation of e.g. pressures.

2.2 Free surface hydraulics

So far, bulk flow schemes have been described. For the application to free surface flow problems, different modifications have to be made. For shallow water and Bernoulli equations, these modifications are minor, as these model equations were derived under the assumption of certain water heights and pressure potentials. The remaining equations have to be combined with proper boundary conditions to assure a valid treatment of the free surface. Kinematic and dynamic boundary conditions have to be distinguished. Kinematic free surface boundary conditions (often referred to as KFSBC) are restrictions on the water particle kinematics. On any interface, at a solid boundary or the air-water interface, the particle velocity has to equal the interface velocity. This is very clear for solid boundaries, as a special case of an interface boundary, where the kinematic boundary condition yields a no flow boundary condition. The fluid velocity normal to the interface has to be equal to the interface normal velocity. Similarly, at the free surface boundary, the interface F has to move with particle velocity:

$$\mathbf{u}\cdot\mathbf{n} = -\frac{\partial_t F}{\nabla F} \tag{2.14}$$

with the unit surface normal $\mathbf{n} = \nabla F/|\nabla F|$. Secondly, in addition to a consistent interface movement, the free surface dynamics are also related to pressure and have to fulfill a certain pressure balance along the interface. This BC often is referred to as *dynamic free surface boundary condition* (DFSBC in literature) and prescribes the pressure at the interface:

$$p = p_0 \tag{2.15}$$

In Navier-Stokes and Euler-based free surface flow solvers, the kinematic boundary condition is usually fulfilled by the advection scheme, whereas the dynamic boundary condition is applied similarly to a pressure boundary condition. In potential flow theory, which is solely based on the flow kinematics, a recourse to Bernoulli equations is needed to introduce the pressure. Usually, a truncated, linearized Taylor series is used for the advection scheme, which is extended to second-order in more sophisticated models.

2.2.1 Water wave theories

Apart from the previously introduced bulk flow considerations, special theories and theorems have been developed, especially for free surface flow hydraulics, concentrating on the propagation of long waves and the typical flow fields below progressive ocean waves. These theories should be reviewed or at least be known before dealing with free surface flow problems and numerical implementations of complex three-dimensional free surface flow simulation schemes. As a minimum, they can provide meaningful validation test cases, as for some test cases, analytical or semi-analytical solutions exist. Dean and Dalrymple [27] give an elaborate overview of (mostly linear) water wave mechanics.

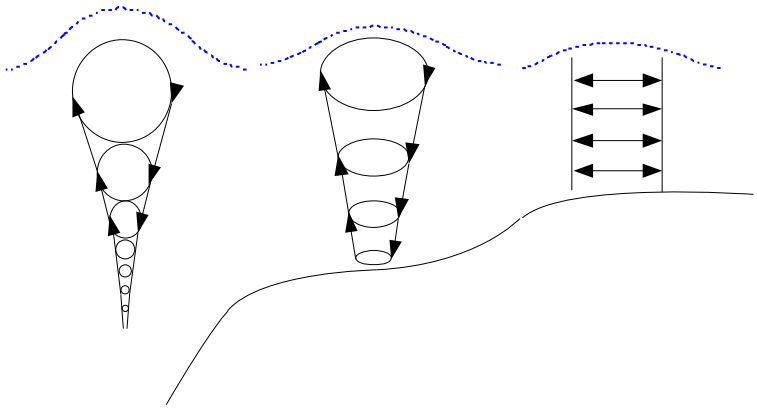


Figure 2.1: Deep, intermediate and shallow water regimes (following Dean and Dalrymple [27])

Technically speaking, ocean waves are gravity waves, as gravity represents the main restoring force balancing inertia when they propagate on the ocean surface. Indeed, the gravitational force always tries to restore the state of minimal energy, i.e. the state of a flat surface with the same potential everywhere. Different, and more or less complicated theories can be used to describe wave motion. In linear wave theory (a.k.a., Airy wave theory, Airy [6]), a linearized description of free surface conditions is used. This theory is based on potential flow theory and therefore assumes that the fluid flow is inviscid, incompressible and irrotational. Linear wave theory is widely used in ocean engineering. It provides a description of wave kinematics and also the wave dynamics, which is of first order, but sufficiently accurate for many purposes, particularly on deep water. Linear theory is often used to get a quick and rough estimate of wave characteristics and their effects. The surface elevation below a progressive wave in a fluid of a uniform mean depth is given by

$$\eta(x,t) = a\cos(kx - \omega t) \tag{2.16}$$

with wave amplitude a, wave number k and frequency ω. The corresponding potential, which fulfills linearized kinematic and dynamic boundary conditions reads

$$\Phi = \frac{\omega}{k} a \frac{\cosh\big(k(z+h)\big)}{\sinh(kh)} \sin(kx - \omega t) \tag{2.17}$$

for the most general case of intermediate water. The linear dispersion relation, which is required for the existence and uniqueness of the potential, reads

$$\omega^2 = gk \tanh(kh). \tag{2.18}$$

For the limit of deep and shallow water, the analytical solutions can be further simplified, due to the asymptotic behavior of the hyperbolic functions.

Airy wave theory is sufficient for modeling the main motions of deep-water waves. In reality, the swell shapes in the ocean are not perfectly sinusoidal, but exhibit a more complex but still periodic shape. This can be taken into account by adding additional terms to Eq. 2.16. The wave theory also allows for second-order effects, such as Stokes drift, and more complex wave shapes, as can be seen in Fig. 2.2.

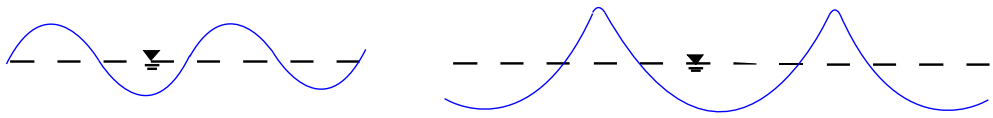


Figure 2.2: Linear and second-order wave theory

2.2.2 Boussinesq approximation and Korteveg-de-Vries (KdV) equation

In addition to linear and non-linear wave theories, the Boussinesq and Korteweg-de Vries (KdV) equations provide a powerful tool for modeling weakly non-linear and dispersive long waves [28], and should be mentioned for completeness. Even if these are not further used in this thesis, their relation to the shallow water equations is of great importance for the analysis and benchmarking the SWE models.

In the Boussinesq equations, similar to the shallow water equations, the vertical coordinate is eliminated from the governing equations. Instead of a straightforward depth-averaging (as in shallow water models), some of the vertical flow structure is conserved, which amounts to keeping the non- hydrostatic effects of vertical accelerations in the equations. Basically, a truncated Taylor expansion up to a certain order in combination with continuity equation and the irrotational-flow-assumption serves to eliminate vertical partial derivatives from the momentum equation. Boussinesq [12] formulates different sets of equations. Set *A* reads

$$\partial_t \eta + \nabla\left[(h+\eta)\mathbf{v}\right] = \frac{1}{6} h^3 \nabla^3 \mathbf{v} \tag{2.19}$$

$$\partial_t \mathbf{v} + \mathbf{v}\nabla\mathbf{v} + g\nabla h = \frac{1}{2} h^2 \partial_t \nabla^2 \mathbf{v} \tag{2.20}$$

Neglecting the dispersive terms on the right-hand sides of these equations leads to the shallow water equations. The main difference between those two sets of equations is thus the frequency dispersion.

A Boussinesq-type model can be used for relatively small relative wave lengths (in relation to the mean water depth). The non-dispersive shallow water equations have a higher relative error and the wave lengths under consideration have to be much longer (typically at least 20 times) than the water depth. With the additional simplification of forward only propagation (i.e., no reflection is possible), the Boussinesq equations reduce to the KdV equation which model wave propagation in one horizontal dimension. Solutions include prototypical examples of solitons. The KdV equation is a nonlinear, dispersive partial differential equation:

$$\partial_t \phi + \partial_x^3 \phi + 6\,\phi\,\partial_x \phi = 0. \tag{2.21}$$

Solutions of both Boussinesq and KdV equations include prototypical examples of solitons. Besides solitary wave solutions, these equations also have exact periodic solutions, called cnoidal waves. The cnoidal wave solutions were derived by [96].

2.2.3 Breaking waves classification

Wave theories can be used to describe wave trains during propagation in shallow, intermediate, or deep water. During wave runup onto plane beaches, the water height decreases, leading to a transition from deep to intermediate and potentially to shallow water conditions, depending on wave length and bottom slope. This also affects the phase speed, as predicted by the wave dispersion relation. The wave crest and the wave steepness become steeper and steeper, finally leading to wave breaking. According to Galvin [44], four types of wave breaking can be identified (Fig. 2.3):

- spilling breaker: white foam appears at the wave crest and spills down the front face of the wave, typically found on flat beaches
- plunging breaker: as the wave steepens, a plunging jet develops and finally impacts on the water surface
- collapsing breaker: the crest of the steepening wave remains unbroken and the lower part of the front face explodes
- surging breaker: the wave does not break, but moves up and down the beach, usually found on very steep beaches

The transition between these four wave breaking regimes is smooth. Especially plunging breakers are of high interest for engineering applications, as the plunging jet may impact on structures. At the same time, this test case is very demanding for the free surface tracking scheme. In section 8.3, a plunging breaker during shoaling is examined.

2.3 Dimensionless quantities

Before starting with numerical modeling, one last topic has to be addressed in order to guarantee accurate simulations. Dimensional analysis is one of the most important fields in fluid dynamics, as it allows identifying dimensionless parameters that characterize a flow state or a flow problem. Even if scales are changed, the dimensionless parameters have to be the same to ensure dynamical flow similarity, which is of particular importance when dealing with scale models in experimental or

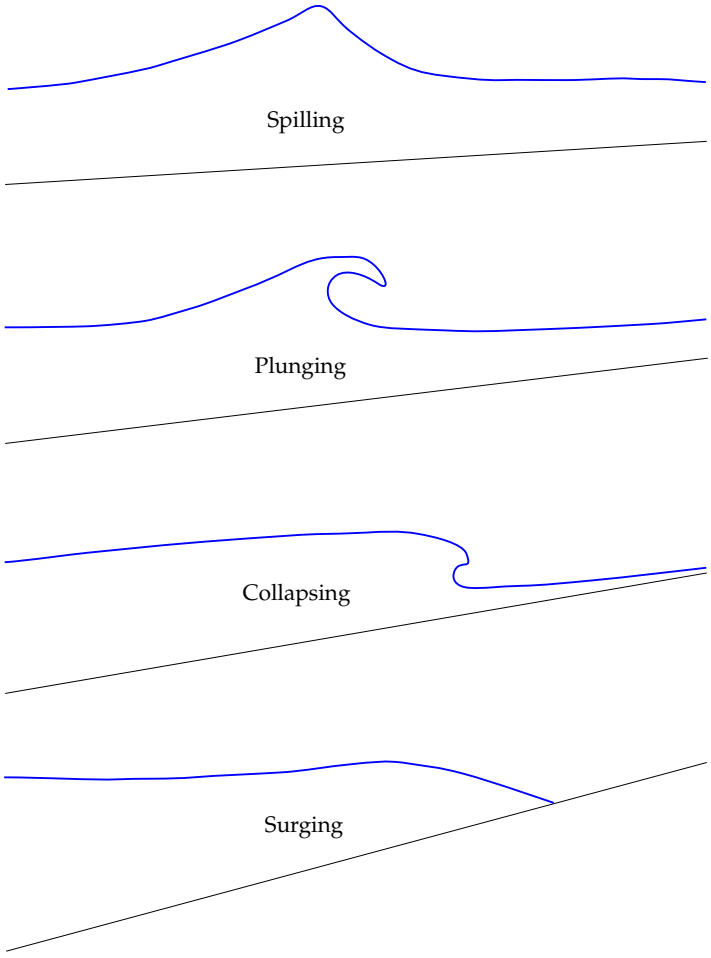


Figure 2.3: Breaker type classification (following Galvin [44])

numerical wave tanks. In experimental work, the existing physical properties of the real world flow problems often can not be expressed one by one. Reduced scales for space and time have to be chosen. To be able to capture the same basic effects, the remaining free model parameters have to be properly adjusted. For experimental tests, this is not possible due to limitations in available fluid viscosity and density, and as a change of earth gravity is quite complicated to achieve. As a consequence, if numerical results are compared to experimental data, the numerical simulations should be run at the experimental length scale instead of the real-world one. Numerical methods are in the convenient position of changing flow properties (e.g., density, gravity, speed of sound, viscosity) to nearly arbitrary values. In turn, numerical methods are limited in terms of grid resolution and effects of turbulence.

In the following, the four most important dimensionless quantities for flow problems studied in this thesis are briefly discussed.

Reynolds number The most widely used dimensionless parameter in fluid dynamics is the *Reynolds number*. It quantifies the relation between inertia and viscous forces. The Reynolds number is defined

as

$$\mathrm{Re} = \frac{v_\infty \cdot L_0}{\nu} \tag{2.22}$$

with characteristic velocity v_∞, characteristic length L_0 and kinematic viscosity ν. For low Reynolds numbers viscous effects dominate the flow behavior and damp developing fluctuations. At high Reynolds numbers, fluctuations appear. In CFD, these fluctuations demand special care, such as the use of turbulence models and/or highly refined computational grids to capture the small scale fluctuations properly. Reynolds numbers of typical free surface flow problems in coastal engineering are in the range of 10^6 to 10^8.

Froude number For free surface flows, the Froude number Fr is the second fundamental dimensionless parameter. It is defined as the ratio of inertial to gravitational forces, namely

$$\mathrm{Fr} = \frac{v}{\sqrt{gh}}\,, \tag{2.23}$$

with characteristic velocity v, characteristic length h and gravity g. It characterizes the nature of the flow to be super- or subcritical. In subcritical flows ($\mathrm{Fr} < 1$), the flow velocity is lower than the long wave velocity $c = \sqrt{gh}$, in supercritical flows ($\mathrm{Fr} > 1$) it is higher. The equivalent in gas dynamics is the Mach number.

Mach number The Mach number is of great importance in acoustic-dominated problems. It qualifies the ratio of flow speed v to the speed of sound in the medium of sound transmission c_s:

$$\mathrm{Ma} = \frac{v}{c_s} \tag{2.24}$$

Typical values for c_s are 343.26 m/s (air) and 1560 m/s (salt water). For low Mach numbers, the ideal gas laws can be used and nonlinear compressibility effects can be neglected. For typical large-scale free surface flow problems, due to the high speed of sound in water, the Mach number is of minor importance.

Knudsen number As incipiently mentioned, the continuum hypothesis is the basis of all macroscopic Navier-Stokes-based approaches. The Knudsen number is useful to qualify whether statistical mechanics or the continuum mechanics formulation of fluid dynamics should be used. It is defined as the ratio of the molecular mean free path length to a certain representative physical length scale. This length scale could be, for example, the radius of a body in a fluid, and can be related to the Mach and Reynolds number of a flow problem according to

$$\mathrm{Kn} = \frac{\lambda}{L} = \frac{Ma}{Re}\sqrt{\frac{\gamma\pi}{2}} \tag{2.25}$$

where γ is the heat capacity ratio $\gamma = \frac{C_P}{C_V}$. If the Knudsen number is near or greater than unity, the mean free path of a molecule is comparable to a length scale of the problem, and the continuum hypothesis of fluid mechanics is no longer a valid approximation. In this case, statistical methods have to be used. For large scale free surface flow problems at low Mach numbers and high Reynolds numbers, which are the main applications of this thesis, the Knudsen number is very low.

CHAPTER 3

The Lattice Boltzmann Method

The Lattice Boltzmann method (LBM) is a relatively new simulation technique for computational fluid dynamics, which has matured to an alternative to classical CFD solvers based on Navier-Stokes equations. It has its origins in the kinetic gas theory, and early implementations of its ancestors (the lattice gas automata) already attracted people's attention in the early 1970s and 1980s. For a deeper understanding, a short review of kinetic methods and a short derivation of the LBM will be given. For further details, the interested reader is referred to the work of [152, 97, 161, 176].

3.1 Kinetic methods

By contrast with the widely known CFD solvers, kinetic methods regard flow problems on a microscopic scale instead of solving the macroscopic Navier-Stokes equations. They are based on the kinetic theory of gases, with the basic assumption that gases consist of a large number of particles in statistically constant, random motion. Hence, the macroscopic properties such as viscosity and pressure are explained by molecular composition and motion.

The first numerical realization of kinetic gas theory goes back to the Lattice gas automata (LGA) of Hardy et al. [72] and Frisch et al. [43]. On either a square lattice (HHP model) or a hexagonal lattice (FHP model), the motion of single particles is tracked, see Fig. 3.1. A Boolean variable indicates if a particle is moving in one discrete lattice direction or not. Particle motion is controlled by collision rules and propagation rules. At the boundaries, particles bounce off or forward. All particles have the same mass, and are only allowed to travel to their nearest neighbor location in one time step. FHP models were already able to reproduce solutions of the Navier-Stokes equations.

After the high initial expectations, more and more drawbacks were discovered. LGAs suffer from a lack of Galilean invariance, statistical noise and a limited, relatively low, maximum Reynolds number. Moreover, the exponential complexity of the collision operator, in combination with the limited

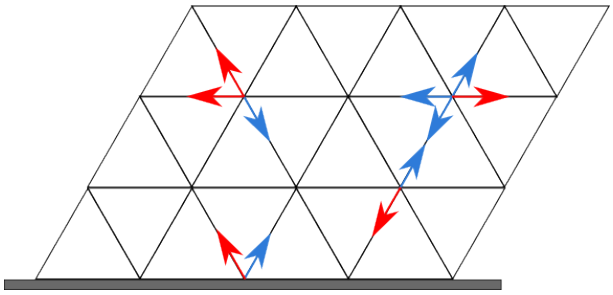


Figure 3.1: FHP Lattice

Reynolds number, diluted the interest in LGAs for complex, three-dimensional high Reynolds number, simulations and pioneered the Lattice Boltzmann Method (LBM).

3.2 Basics of LBM

The basic idea of LBM is to replace the Boolean particle number with its ensemble-average and to introduce a particle distribution function $f(t, \mathbf{x}, \boldsymbol{\xi})$, which specifies the probability to encounter molecules at position $\mathbf{x}$ at time t with velocity $\boldsymbol{\xi}$. The first LBE goes back to McNamara and Zanetti [113]; improvements were made by Higuera and Jiménez [80], Higuera et al. [81].

In 1872 Ludwig Boltzmann derived an equation describing the development of these distribution functions f due to interactions on the microdynamic scale:

$$\frac{Df}{Dt} = \frac{\partial f(t,\mathbf{x},\boldsymbol{\xi})}{\partial t} + \boldsymbol{\xi} \cdot \frac{\partial f(t,\mathbf{x},\boldsymbol{\xi})}{\partial \mathbf{x}} + \mathbf{F} \cdot \frac{\partial f(t,\mathbf{x},\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} = \Omega \tag{3.1}$$

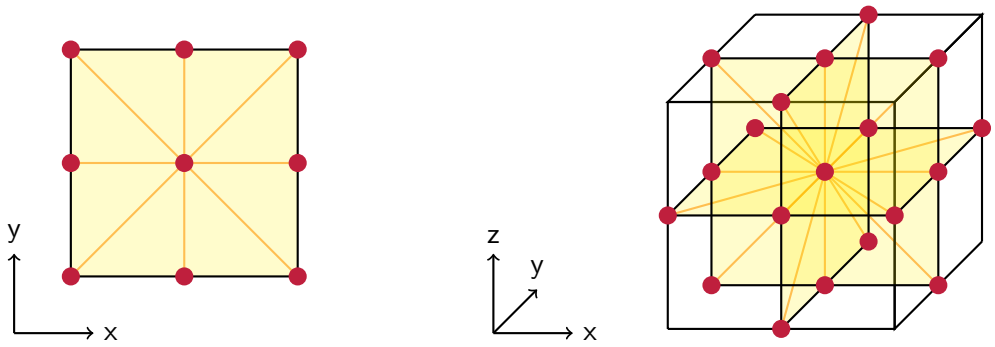
dealing with the total differential of f, which is further transformed using the relations $\xi = \partial_t \mathbf{x}$ and $\mathbf{F} = \partial_t (m\boldsymbol{\xi}) = \partial_t \boldsymbol{\xi}$. The external forces $\mathbf{F}$ are neglected in the remainder of this derivation and are reintroduced in section 3.5.

The left hand side of the Boltzmann equation is of advection type, while the right hand side contains the *collision operator* Ω, which describes the interaction of particles. The collision integral is a complex non-linear integral expression and leads to an overall integro-differential equation which is hard to solve. Simplifications, such as the BGK collision operator, have been proposed, which basically display similar collision properties as the complex integral, but are drastically easier to solve. Details of the Boltzmann equation and different continuous collision operators can be found in Succi [152]. However, the Boltzmann equation, even with simplified collision operator, is a partial differential equation in space, time and velocity space.

For continuum flows with low Knudsen numbers, discretized velocities $\mathbf{e}_i$ may be introduced in order to obtain a model with reduced computational costs. A particle is only allowed to move in a limited number of directions and specific distances, so that Eq. 3.1 can be transformed to a set of discrete Boltzmann equations:

$$\frac{\partial f_i(t,\mathbf{x})}{\partial t} + \mathbf{e}_i \frac{\partial f_i(t,\mathbf{x})}{\partial \mathbf{x}} = \Omega_i \tag{3.2}$$

In this work the D3Q19 model is used for the discretization of the velocity space [134]. Several other models are available for the hydrodynamic purpose, such as e.g. the D3Q13, D3Q15 or D3Q27 model, and the D2Q6, D2Q7 and D2Q9 models for two spatial dimensions. If higher Mach number flows or flows at high Knudsen numbers are targeted, even more complex velocity discretization stencils have to be used, such as the D2Q17 or D3Q39 model, see e.g. Shan et al. [146].



The velocity space of the D3Q19 model introduces 19 velocities

$$\{\vec{e}_i, i = 0,\dots,18\} = c \cdot \left\{ \begin{array}{c|cccccc|cccccccccccc} 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \end{array} \right\} \tag{3.3}$$

whereas the velocity space of the D2Q9 model reads

$$\{\vec{e}_i, i = 0,\dots,8\} = c \cdot \left\{ \begin{array}{c|cccccc|cc} 0 & 1 & -1 & 0 & 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 \end{array} \right\}, \tag{3.4}$$

with a constant velocity c, determining the speed of sound $c_s = c/\sqrt{3}$

A finite difference discretization in space and time on a grid with $c = \Delta x/\Delta t = 1$ (grid spacing Δx, time stepping Δt) leads to the *Lattice* Boltzmann equation

$$f_i(t + \Delta t, \mathbf{x} + \mathbf{e}_i \Delta t) - f_i(t, \mathbf{x}) = \Omega_i \tag{3.5}$$

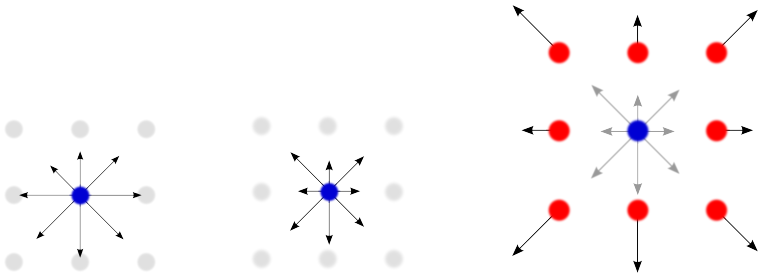
Finally, Eq. 3.5 may be split up into a local *collision* step which drives the particle distribution functions to equilibrium locally

$$\bar{f}_i(t, \mathbf{x}) = f_i(t, \mathbf{x}) + \Omega_i \tag{3.6}$$

and a *propagation* step, where the evolved particle distribution function is moved to the respective neighbor

$$f_i(t + \Delta t, \mathbf{x} + \mathbf{e}_i \Delta t) = \bar{f}_i(t, \mathbf{x}). \tag{3.7}$$

The locality of the nonlinear collision operations and the linearity of the advection step allow for very efficient implementations on high-performance hardware, such as massively parallel PC clusters and GPUs. See the performance analysis of our GPU implementation in chapter 5.



(a) Collision (b) Propagation

Figure 3.2: Collision and propagation

The well known macroscopic quantities pressure and velocity are related to the first two hydrodynamic moments of the distribution functions for density fluctuation ρ and momentum $\rho_0 \mathbf{v}$

$$p = \rho c_s^2 = c_s^2 \sum_{i=0}^{18} f_i \quad \text{and} \quad \mathbf{v} = \frac{1}{\rho_0} \sum_{i=0}^{18} \mathbf{e}_i f_i \tag{3.8}$$

It can be shown that the solutions of Lattice Boltzmann equation fulfill the incompressible Navier-Stokes equations up to certain errors. The Chapman-Enskog expansion or asymptotic analysis (see Junk et al. [91]) can be used to this effect. However, the Lattice Boltzmann equation reproduces solutions of the following macroscopic equations:

$$\nabla(\mathbf{v}) = 0 + \mathcal{O}(\Delta t^2) + \mathcal{O}(\mathrm{Ma}^2) \tag{3.9}$$

$$\partial_t \mathbf{v} + (\mathbf{v}\nabla)\mathbf{v} = -\frac{1}{\rho_0}\nabla p + \nu \Delta \mathbf{v} + \mathcal{O}(\Delta t^2) + \mathcal{O}(\mathrm{Kn}^2) + \mathcal{O}(\mathrm{Ma}^2) \tag{3.10}$$

which correspond to the incompressible Navier-Stokes equations up to discretization errors of $\mathcal{O}(\Delta t^2)$, a Knudsen error of $\mathcal{O}(\mathrm{Kn}^2)$ and a compressibility error of $\mathcal{O}(\mathrm{Ma}^2)$. It is also notable that the pressure is obtained as a zeroth order moment from the particle distribution functions. No additional and cumbersome Poisson equation has to be solved.

3.3 Discrete collision operators

In the Boltzmann equation, the collision operator Ω denotes a complex integral operator, modeling the interactions of two particles. Even in the continuous formulation, several approaches are available to simplify the collision operators. The same holds for the discrete version of the Boltzmann equation. In the following, SRT and MRT models will be presented. Both collision operators have in common that they express a relaxation-like process to an equilibrium state. The equilibrium state is linked to the state of a stress-free fluid. Hence, the relaxation process is coupled to the fluid viscosity ν.

3.3.1 Single relaxation time model

A simple approximation for the collision term Ω_i is the single relaxation time (SRT) approximation developed by Bhatnager, Gross and Krook (BGK) [10]:

$$\Omega_i = -\frac{\Delta t}{\tau}(f_i - f_i^{eq}). \tag{3.11}$$

Here $\tau = 3\nu + \frac{1}{2}\Delta t$ is the relaxation time coupled to the kinematic viscosity ν and f_i^{eq} is a low order polynomial approximation of the Maxwellian equilibrium distribution [18] :

$$f^{eq} = \frac{\rho}{(2\pi c_s^2)^{d/2}} \exp -\frac{\xi^2}{2c_s^2} \tag{3.12}$$

The discrete equilibrium distributions [143] are in general calculated via

$$f_i^{eq} = w_i \rho \left(1 + 3\,\frac{\mathbf{e}_i \cdot \mathbf{u}}{c^2} + \frac{9}{2}\frac{(\mathbf{e}_i \cdot \mathbf{u})^2}{c^4} - \frac{3}{2}\frac{u^2}{c^2}\right). \tag{3.13}$$

The equilibrium distribution functions tuned for incompressible flows [78] are

$$f_i^{eq} = w_i \left(\rho + \rho_0(3\,\frac{\mathbf{e}_i \cdot \mathbf{u}}{c^2} + \frac{9}{2}\frac{(\mathbf{e}_i \cdot \mathbf{u})^2}{c^4} - \frac{3}{2}\frac{u^2}{c^2})\right) \tag{3.14}$$

where ρ_0 is the reference density, ρ is a density variation and w_i are weighting factors depending on the chosen discretization model. For the D3Q19 model these read

$$w_0 = \frac{1}{3}, \quad w_{1..6} = \frac{1}{18} \quad \text{and} \quad w_{7..18} = \frac{1}{36}. \tag{3.15}$$

3.3.2 Multiple relaxation time model

In the more advanced MRT model [29] the particle distribution functions are transformed into moment space before relaxation. The moments $\boldsymbol{m} = \mathrm{M}\boldsymbol{f}$ are labeled as

$$\mathbf{m} = \left(\rho, e, \epsilon, j_x, q_x, j_y, q_y, j_z, q_z, 3p_{xx}, 3\pi_{xx}, p_{ww}, \pi_{ww}, p_{xy}, p_{yz}, p_{xz}, m_x, m_y, m_z\right).$$

and denote the mass density $m_0 = \rho$, the part of the kinetic energy independent of the density ($m_1 = e$), the part of the kinetic energy square independent of the density and kinetic energy ($m_2 = \epsilon$), the momentum ($m_{3,5,7} = j_{x,y,z}$). $m_{4,6,8} = q_{x,y,z}$ are related to heat flux, $m_{9,11,13,14,15}$ are related to the symmetric traceless viscous stress tensor, $m_{16,17,18}$ are third-order moments and $m_{10,12}$ are two fourth order moments.

Several different relaxation rates are used. This leads to an increase in stability and allows the development of more accurate boundary conditions and thus more efficient simulations, see e.g. Ginzburg and D'Humieres [53] . The collision operator for MRT reads

$$\Omega_i = \mathbf{M}^{-1}\mathbf{S}(\mathbf{M}f - m_i^{eq}) \tag{3.16}$$

$\mathbf{M}$ denotes the transformation matrix from distribution functions to moment space ($m_i = \mathbf{M}f$ and $f = \mathbf{M}^{-1}m_i$) and is given in Appendix A. m_i^{eq} are the equilibrium moments given by

$$m_0^{eq} = \rho, \quad m_3^{eq} = \rho_0\, u_x, \quad m_5^{eq} = \rho_0\, u_y, \quad m_7^{eq} = \rho_0\, u_z, \tag{3.17a}$$

$$m_1^{eq} = e^{eq} = \rho_0\,(u_x^2 + u_y^2 + u_z^2), \tag{3.17b}$$

$$m_9^{eq} = 3p_{xx}^{eq} = \rho_0\,(2\,u_x^2 - u_y^2 - u_z^2), \tag{3.17c}$$

$$m_{11}^{eq} = p_{zz}^{eq} = \rho_0\,(u_y^2 - u_z^2), \tag{3.17d}$$

$$m_{13}^{eq} = p_{xy}^{eq} = \rho_0\, u_x\, u_y, \tag{3.17e}$$

$$m_{14}^{eq} = p_{yz}^{eq} = \rho_0\, u_y\, u_z, \tag{3.17f}$$

$$m_{15}^{eq} = p_{xz}^{eq} = \rho_0\, u_x\, u_z, \tag{3.17g}$$

where ρ_0 is a constant density and ρ a density variation. The velocities are derived from the moments representing momenta: $u_\alpha = j_\alpha/\rho_0$. S $= s_{i,i}$ is the diagonal collision matrix, which contains the relaxation parameters

$$\begin{aligned}
s_{1,1} &= s_a \\
s_{2,2} &= s_b \\
s_{4,4} &= s_{6,6} = s_{8,8} = s_c \\
s_{10,10} &= s_{12,12} = s_d \\
s_{9,9} &= s_{11,11} = s_{13,13} = s_{14,14} = s_{15,15} = -\frac{\Delta t}{\tau} = s_\omega \\
s_{16,16} &= s_{17,17} = s_{18,18} = s_e.
\end{aligned} \tag{3.18}$$

The unmentioned relaxation parameters are set to zero. The relaxation time τ is

$$\tau = 3\frac{\nu}{c^2} + \frac{1}{2}\Delta t, \tag{3.19}$$

where ν is the kinematic viscosity. The parameters s_a, s_b, s_c, s_d and s_e can be freely chosen in the range $[-2, 0]$ and tuned to improve stability [102]. While the optimal values for these parameters depend on the specific system under consideration (geometry, initial and boundary conditions), reasonable values are given in [29]. We choose $s_a = s_b = s_c = s_d = s_e = -1.0$.

3.3.3 Advanced models

For sake of completeness, more sophisticated and specialized discrete collision operators should be at least mentioned.

In the two-relaxation-time (TRT) LB model going back to Ginzburg et al. [56], two different relaxation times are used. In a very simple linear collision operator the particle distribution functions are decomposed into symmetric and anti-symmetric components, which then are relaxed independently. If both relaxation times are identical, the TRT operator reduces to the BGK SRT collision operator. In a similar way, TRT collision can be reproduced with an MRT collision operator by selecting proper relaxation times.

The cascaded Lattice Boltzmann Method aims at stable collision operators even in the limit case of very low viscosity. SRT and MRT models suffer instabilities in the low viscosity limit, which is - among other reasons - due to an insufficient degree of Galilean invariance of the relaxation-type Lattice Boltzmann collision operator. Geier et al. [47] propose a cascaded formulation. On the basis of a full set of velocities (27 discrete speeds), central moments are defined and relaxed in an consecutive, cascaded relaxation process. This model is typically stable for ω close to 2.0, which corresponds to ν close to zero, and all other relaxation parameters set to unity.

3.4 Boundary conditions

The solution of partial differential equations requires the specification of boundary conditions at the domain boundaries in order to obtain a unique solution of the problem under consideration. In the LBM, boundary conditions have to be directly specified for the distribution functions of the boundary nodes, which is quite different from macroscopic CFD methods. The most common boundary conditions will be shortly presented in the following sections.

3.4.1 No slip boundary conditions

At solid boundaries, the fluid sticks to the solid surface, which is called a no-slip boundary condition. Physically this takes into account that the first fluid particle in the domain does not move. The velocity component normal to the impermeable wall has to be zero anyway. In the LB context, no-slip wall boundary conditions are modeled with *bounce-back schemes*. Particles in the vicinity of the boundary hit the obstacle and bounce off the wall again. In a simple (first order) bounce-back scheme, the particles bounce back in the direction they came from. This way, the total fluid velocity is approximately zero (according to Eq. 3.8), and momentum is exchanged between the fluid and solid, as the particles change direction. The first-order bounce-back scheme does not consider higher-order geometry information and is only second-order accurate in space, if the solid obstacle is located in the center of two neighboring lattice nodes.

To improve the accuracy and to be able to cope with more complex and non-axis-parallel geometries, Bouzidi et al. [13] propose a second-order bounce-back scheme. It slightly violates conservation of mass, but is second order in space, so that an overall scheme of second order accuracy may be achieved. Attempts to resolve this mass loss have been performed by [4], adding the mass difference to the resting particle distribution. However, this method induces higher-order disturbances to the stress tensor without showing significant impact on the results. The modified bounce back scheme reads

$$f_{inv,A}^{t+1} = \begin{cases} (1-2q)\, f_{i,B}^{t} + 2q\, f_{i,A}^{t} - 2\rho w_i \dfrac{\mathbf{e}_i \cdot \mathbf{v_w}}{c_s^2}, & 0.0 < q < 0.5, \\ \dfrac{(2q-1)}{2q}\, f_{inv,A}^{t} + \dfrac{1}{2q}\, f_{i,A}^{t} - \rho w_i \dfrac{\mathbf{e}_i \cdot \mathbf{v_w}}{q c_s^2}, & 0.5 \leq q \leq 1.0. \end{cases} \tag{3.20}$$

f_{inv} and f_i are anti-parallel distributions, and f_{inv} is the incoming (missing) and f_i the outgoing distribution function. q denotes the wall distance for direction i and u_w denotes the wall velocity, i.e. for moving obstacles, see Fig. 3.3. Basically, as particles only travel exactly one grid spacing per time step, per definition, inter- or extrapolation yields the value of the particle distribution function before streaming.

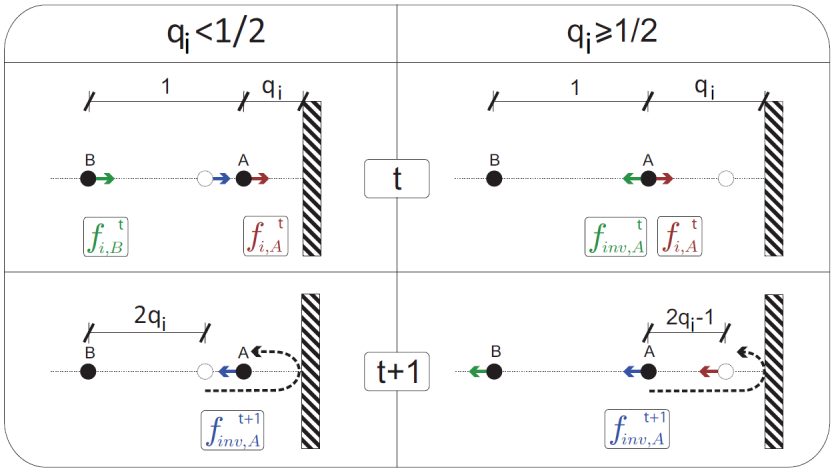


Figure 3.3: Second-order bounce-back scheme (illustration from Freudiger [42])

3.4.2 Slip boundary conditions

The use of no-slip boundaries is sufficient for most CFD applications, but does not make sense, when the fluid is assumed to be inviscid or the flow boundary is assumed to be frictionless. In these two cases, no tangential momentum is exchanged at the solid-fluid interface and the fluid is allowed to move freely. This boundary condition often is referred to as slip boundary condition. Similar to no-slip BCs, the velocity component perpendicular to the non-moving wall still is required to be zero (no flow boundary condition). Especially for free surface flow problems at high Reynolds numbers and at high grid resolutions, the use of slip boundary conditions can be a good approximation. In more advanced approaches, viscous effects in the thin near- wall sublayer are represented by wall functions. In analogy to bounce-*back* schemes, slip boundary conditions can be modeled as bounce-*forward* pattern. Particles are reflected off the boundary, but they bounce forward. To ensure that no momentum is exchanged with the wall, the momentum tangential to the wall is not changed. This simple scheme leads to very efficient and accurate boundary conditions, but suffers from the same drawbacks as the simple bounce-back scheme.

Ahrenholz [5] proposes an enhanced formulation, which uses information about the wall normal vector. The velocity vector at the slip boundary $\mathbf{v}$ is modified by subtracting its projection on the wall surface normal vector $\mathbf{n}$ as

$$\mathbf{v}_{\parallel} = \mathbf{v} - (\mathbf{v} \cdot \hat{\mathbf{n}})\hat{\mathbf{n}} \tag{3.21}$$

and finally applied as boundary velocity $\mathbf{v}_{\parallel}$ using the following modified version of the second-order bounce-back scheme:

$$f_{inv,A}^{t+1} = \begin{cases} (1-2q)\, f_{i,F}^{t} + 2q\, f_{i,A}^{t} - 2\rho w_i \dfrac{\mathbf{e}_i \cdot \mathbf{v}_{\parallel}}{c_s^2}, & 0.0 < q < 0.5, \\ \dfrac{(2q-1)}{2q}\, f_{inv,A}^{t} + \dfrac{1}{2q}\, f_{i,A}^{t} - \rho w_i \dfrac{\mathbf{e}_i \cdot \mathbf{v}_{\parallel}}{q c_s^2}, & 0.5 \leq q \leq 1.0. \end{cases} \tag{3.22}$$

Hence, a frictionless behavior with approximately zero wall shear stress is modeled.

3.4.3 Velocity boundary conditions

Velocity boundary conditions are treated similar to moving obstacle boundary conditions. The second-order bounce-back scheme with a prescribed wall velocity is used.

3.4.4 Pressure and extrapolation boundary conditions

At open boundaries or outflow boundaries, respectively, the flow velocity is often unknown, so that either a pressure boundary condition or extrapolation boundary condition are used. In the literature, several formulations have been proposed for pressure boundary conditions in the LB context. They all have in common that they need information on the macroscopic values for pressure and velocity at the boundary. To obtain these information, either the neighboring values are extrapolated, or - if available - boundary information is specified. In extrapolation boundary conditions, the required macroscopic values are usually obtained by extrapolation from the penultimate fluid node inside the domain. This constant extrapolation in space works fine if the flow direction is clearly defined and the upwinding is valid. As soon as backflow occurs or the pressure gradient in the vicinity of the outflow boundary is not zero, however, the system density will typically increase. Hence, at least the value for the node density should be set to the prescribed boundary pressure, whereas for the velocity information the above mentioned extrapolation method is used. Generally speaking, on at least one point in the system, the reference pressure should be adjusted anyway, in order to obtain a well-posed system with stable and reliable results.

In any case, these macroscopic values serve as input parameters to the actual boundary condition. Three formulations should be mentioned: a very robust and crude method is to specify Maxwellian equilibrium distribution functions at the boundary node at every time step. The boundary condition is based on the assumption that the higher-order moments and the non-equilibrium parts of the particle distribution functions at the boundary are zero, which corresponds to a state of zero stress and is valid in the far-field at outflow boundaries. Secondly, a slightly more sophisticated but still simple approach is to extrapolate the particle distribution functions from the penultimate node in the domain to the boundary node. To avoid unphysical changes in pressure, in a modified extrapolation method, the particle distribution functions are extrapolated to the domain boundary, but then a local pressure correction takes place, modifying the node pressure (i.e. the zeroth order moment) while leaving the higher-order moments unchanged. This corresponds to a zero-gradient assumption for higher order moments, while setting a prescribed boundary density. Thirdly, as most sophisticated pressure boundary condition, Körner et al. [95] propose the anti-bounce back rule given by

$$f_I^{t+1} = -f_i^t + f_I^{eq}(\rho_B, \mathbf{v}(t_B, \mathbf{x}_B)) + f_i^{eq}(\rho_B, \mathbf{v}(t_B, \mathbf{x}_B)). \tag{3.23}$$

$f_{i,I}^{eq}(\rho_B, \mathbf{v}(t_B, \mathbf{x}_B))$ are Maxwellian equilibrium distribution functions where ρ_B is related to the surrounding pressure via $\rho_B = p_B c_s^{-2}$, $t_B = t + \frac{1}{2}\Delta t$ and $\mathbf{x}_B = \mathbf{x} + \frac{1}{2}\hat{\mathbf{e}}_i$. The value for $\mathbf{v}(t_B, \mathbf{x}_B)$ is obtained by extrapolation. As a disadvantage of anti bounce-back, the sensitivity to re-entering vortices has to be mentioned. In this work, the anti-bounce-back boundary conditions are also used at the free surface boundary to fulfill the dynamic free surface boundary condition, leading to an equality of fluid pressure and surrounding atmospheric pressure. It has to be noted that in none of these approaches, the subgrid distance from the fluid node to the actual pressure boundary is taken

into account. Hence the boundary conditions are of $\mathcal{O}(\Delta x)$. Higher-order boundary conditions for the pressure are currently under development.

3.4.5 Non-reflecting boundary conditions

The use of non-reflecting boundary conditions is crucial for various applications. Especially acoustic simulations, which aim at small scale pressure fluctuations, and are sensitive to reflections from the boundaries. In the worst case, the reflected waves overlay the acoustic signal and impurify the simulation results. Izquierdo and Fueyo [87] introduced a LODI (local one-dimensional inviscid equations) ansatz for the treatment of non-reflecting boundary conditions in a LB context. In the LODI ansatz, a one-dimensional set of the inviscid Euler equations is locally solved in the wall normal direction of the boundary. A characteristics-based solution strategy is chosen, which allows for eliminating large portions of the reflected acoustic signal, i.e. the reflected pressure peak. The pressure signal impacting on the wall is relaxed to the prescribed boundary pressure. The resulting values for pressure p_{LODI} and velocity $\mathbf{v}_{LODI}$ are used as input parameters for the classical LB boundary conditions, as given by Eq. 3.20 or Eq. 3.23.

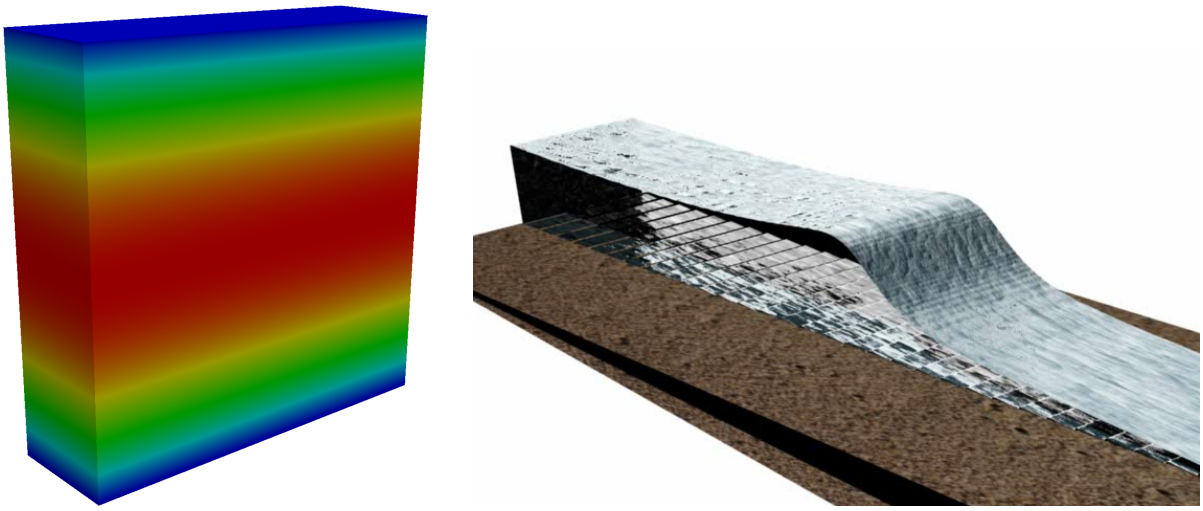
3.4.6 Periodic domains

Periodic boundary conditions serve to simulate unbounded systems in at least one space direction. They are especially useful for the simulation of infinitely long domains, such as a Poiseuille flow between infinite plates (Fig. 3.4a). For free surface flow problems, they serve to simulate quasi-two-dimensional test cases, assuming periodicity in the third direction, e.g. for breaking waves during shoaling (Fig. 3.4b). For explicit Lattice Boltzmann methods, the implementation of periodic boundary conditions is straightforward: the particle distribution functions leaving the domain on one end simply have to be reinserted at the other open end, completing the set of particle distribution functions which was incomplete since the propagation step.

The implementation is straightforward, especially for solvers on the basis of block-structured grids. The standard information exchange method between two neighboring grid blocks can be used to connect one end of a block to the other. Opposite to Navier-Stokes solvers, where periodic BCs are challenging due to jumps in pressure, this is not a problem with the LBM. However, this problem shows up again when pressure discontinuities such as bubbles leave and re-enter, as the pressure gradient is constant but the pressure itself is discontinuous at the periodic boundary. The problem only appears for problems with periodic BCs in direction of the pressure gradient.

3.5 Body forces

Body forces are the driving force of most of the free surface applications, which are mostly dominated by gravitational effects. From the physical point of view, gravity corresponds to a source term in the momentum equation, expressed as a force per unit volume of a body. These source terms are included in the LBM via the *forcing term* F_i, a link-specific contribution of the total body force $\mathbf{F}$. The modified



(a) Poiseuille flow (b) Breaking wave

Figure 3.4: Different applications for periodic boundary conditions

Lattice Boltzmann equation reads

$$f_i(t+\Delta t, \mathbf{x}+\mathbf{e}_i\Delta t) - f_i(t,\mathbf{x}) = \Omega_i + \Delta t F_i \tag{3.24}$$

For the calculation of F_i different approaches are available in literature. In the very straightforward approach of Buick and Greated [15], it is evaluated as

$$F_i = 3\omega_i \rho \mathbf{e}_i \mathbf{F} \tag{3.25}$$

with weight ω_i, density ρ, unit vector $\mathbf{e}_i$ and forcing vector $\mathbf{F}$. Guo et al. [69] compare and analyze different forcing terms and report a lack of accuracy for the former formulation for space- and time-varying body forces $\mathbf{F}$. For these cases, they propose a formulation, which also depends on the relaxation time τ:

$$F_i = \left(1 - \frac{1}{2\tau}\right) \omega_i \left(\frac{\mathbf{e}_i - \mathbf{v}}{c_s^2} + \frac{(\mathbf{e}_i \mathbf{v})}{c_s^4}\mathbf{e}_i\right) \cdot \mathbf{F} \tag{3.26}$$

with the weighting factor ω_i, density ρ and the local fluid velocity $\mathbf{v}$, which is computed with a slightly modified version of Eq. 3.8. Alternatively, in MRT models, the forcing - which is, in fact, a momentum source term - can be directly added to the corresponding momenta.

3.6 Force evaluation

The force $\mathbf{F}$ acting on an obstacle in the flow results from the momentum of the particles hitting the boundary. It can be computed by balancing the particle momentum before and after hitting the boundary:

$$\mathbf{F} = \sum_{i \in \Gamma} \mathbf{F}_i = -\frac{V}{\Delta t}\mathbf{e}_i \left(f_i(t+\Delta t, \mathbf{x}) + f_{inv}(t,\mathbf{x})\right) \tag{3.27}$$

for all links i that are cut by the obstacle and grid cells of volume V [123]. This *momentum exchange method* works well if integral forces on solid obstacles are needed. If the size of the solid obstacle is comparable to the grid spacing, it may lead to relatively poor results, as the quality of the resulting force signal is dependent on the number of links that cut the obstacle. This has been observed by Geller [49] in the coupling of an LB solver to a pFEM structural solver with relatively small single finite elements. To cure the problem, Geller et al. [50] developed a stress integration method, where the stress tensor (which is available locally at the lattice nodes) is integrated along the obstacle surface. For the coupling to rigid bodies, as used in this work, the momentum exchange method is sufficient.

3.7 Turbulence modeling

Free surface flows typically appear at high Reynolds numbers, where turbulent structures contribute to the overall flow pattern. In order to capture turbulent structures in the flow, we use a large eddy model [98]. In LES models, a spatial filter is applied to the velocity field, which should be fine enough not to filter out the large turbulent structures of the flow. Only the effect of the small eddies on the large scale flow structures needs to be modeled and is included in the simulation via an additional turbulent viscosity ν_{T}. We use a Smagorinski model [149] where ν_{T} depends on the shear rate:

$$\nu_{\mathrm{T}} = (C_S \Delta x)^2 \|\mathbf{S}\| \tag{3.28}$$

with Smagorinsky constant C_S, a filter length Δx and the strain rate tensor

$$S_{\alpha\beta} = \frac{1}{2}\left(\frac{\partial v_\alpha}{\partial x_\beta} + \frac{\partial v_\beta}{\partial x_\alpha}\right). \tag{3.29}$$

The strain rate tensor can be computed from the moments

$$S_{\alpha\beta} = \frac{s_{xx}}{2c_s^2\rho}\left(c_s^2\rho\delta_{\alpha\beta} + \rho v_i v_j - P_{\alpha\beta}\right) = \frac{s_{xx}}{2c_s^2\rho} Q_{\alpha\beta} \tag{3.30}$$

with speed of sound c_s, Dirac delta function δ, density ρ, velocity u and the second-order moments P which can be locally computed from $m_{9,11,13,14,15}$. With Eq. 3.30 and

$$\tau_{\mathrm{total}} = \frac{3}{c^2}\nu_{\mathrm{total}} + \frac{1}{2}\Delta t = \frac{3}{c^2}(\nu_0 + \nu_{\mathrm{T}}) + \frac{1}{2}\Delta t \tag{3.31}$$

one obtains a quadratic equation which yields

$$\tau_t = \frac{1}{2}\left(\sqrt{\tau_0^2 + 18C_S^2\Delta x^2\|\mathbf{Q}\|} - \tau_0\right) \tag{3.32}$$

and a modified relaxation rate s_{xx} for the second order moments $m_{9,11,13,14,15}$

$$s_{xx} = \frac{1}{\tau_{total}} = \frac{1}{\tau_0 + \tau_t}. \tag{3.33}$$

A disadvantage of Smagorinsky LES models is the overestimation of the turbulent viscosity in the vicinity of solid boundaries. For future work, advanced large eddy models such as an LES WALE approach (wall-adaptive large eddy simulation, [124]) or a dynamic Smagorinsky model [51, 106] should be considered. Weickert et al. [171] compares different turbulence models in a LB context and focusses on a LES WALE approach and benchmarking of the above-mentioned three models.

3.8 Initial conditions

Apart from proper conditions at the domain boundary, valid initial conditions have to be specified. They have to fulfill the boundary conditions and to represent a valid state of the fluid. Hence, pressure initialization is crucial for flows including effects of gravity. For simulations starting from a state of rest, the particle distribution functions are initialized with Maxwellian equilibrium distribution functions f^{eq}, for a zero velocity field and a corresponding hydrostatic pressure distribution:

$$f = f^{eq}(\rho, \mathbf{v}) \tag{3.34}$$

where ρ is linked to the hydrostatic pressure via $\delta\rho = \delta p c_s^{-2}$, i.e. $\delta\rho = 3\delta\rho g h$ with forcing g and distance to the zero-pressure level h.

The Maxwellian equilibrium state refers to a state with zero higher-order moments and zero non-equilibrium parts. This assumption is not entirely valid, if nonzero initial velocity fields are given. In the latter case, the non-equilibrium parts can be improved by using a local Poisson-type iteration. This idea has been initially proposed by Mei et al. [114]. For a given initial velocity field, the pressure is iteratively improved until it converges to a fixed value. The standard collision and propagation steps are fulfilled, but with a modified version of the collision operator, where the macroscopic velocity is fixed to the prescribed value $\bar{\mathbf{v}}$ and the density is allowed to change:

$$\Omega_i = -\frac{\Delta t}{\tau}\left(f_i - f_i^{eq}(\rho, \bar{\mathbf{v}})\right) \quad \text{and} \quad \Omega_i = \mathbf{M}^{-1}\mathbf{S}(\mathbf{M}f - m_i^{eq}(\rho, \bar{\mathbf{v}})) \tag{3.35}$$

for the SRT and MRT model, respectively. Eventually, the density converges to the correct nodal density consistent with the continuity equation. Mei et al. [114] present convergence studies for a Taylor-Green vortex in a square box, where the analytical solution for both velocities and pressure is known. For MRT models, it has to be noted that during the iteration process, the first-order moments, which are related to the initial velocity, can also be relaxed. This way, the resulting system is less stiff and the iteration shows better convergence properties. The corresponding relaxation factor s_λ can be chosen arbitrarily, whereas correlating s_λ to the local Mach number has shown to produce the best results. SRT models do not have this option.

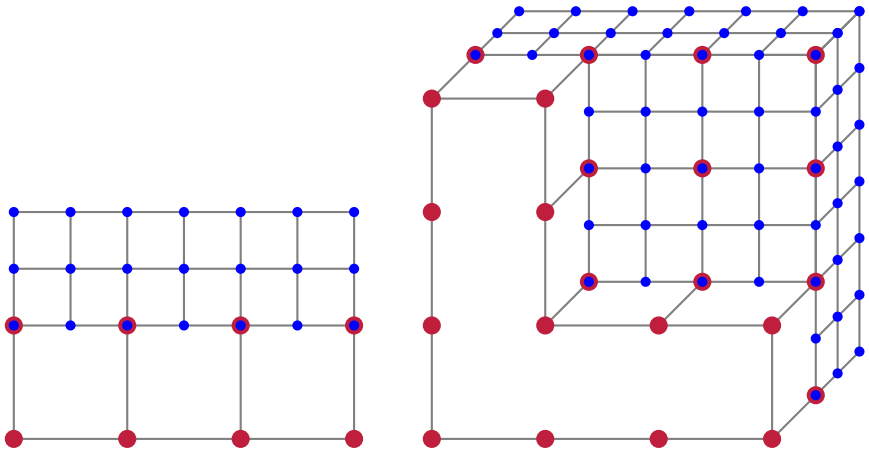
The same iterative method is used for the initialization of new single fluid nodes, which do not carry any valid particle distribution functions. Such new fluid nodes can appear in fluid-structure interaction problems, where Lagrangian solid obstacles sweep over the Eulerian LBM lattice, leading to changes of node states, or for free surface flow simulation, where the interface moves along the lattice, leading to the same effects. Here, the iteration minimizes pressure waves induced by incorrect density initialization.

3.9 Approaches to grid refinement

Non-uniform grids for CFD applications are desirable for various reasons, even today, where computational power is widely available. It makes sense to reduce the grid resolution in the peripheral regions of the flow field, whereas in the main regions of interest and the very sensitive regions such as boundary layers or phase interfaces, a high resolution is preferable. Moreover, the representation of the flow geometry also gains accuracy in the highly resolved grid regions.

During the derivation in section 3.2, the discrete Boltzmann equation has been discretized in space and time with a classical finite difference method. Grid spacing Δx and time step Δt have been coupled via a reference velocity $c = \Delta x / \Delta t$, which is related to the discrete set of discrete particle velocities. Shortly, a particle distribution function is exactly advected one grid spacing in each time step. Hence, to avoid undesired and time-consuming interpolations, the grid spacing should be the same in one subdomain. It has to be pointed out that for different discretization techniques of the (velocity-)discrete Boltzmann equation a non-equidistant space discretization is possible, whereas for an efficient FD discretization, a different grid refinement strategy should be selected. Basically, different subdomains with different space resolutions are coupled and exchange information at the grid interface, resulting in a nested time stepping scheme.

This extensions of the LBM to non-uniform Cartesian grids has been discussed in the literature [21, 22, 38, 179]. Recently, Freudiger [42] and Geller [49] developed the three-dimensional, non-uniform, block-structured LBM solver VirtualFluids with adaptive grid refinement. A typical non-uniform grid transition for a 2D LBM (on the basis of VirtualFluids) is shown in Fig. 3.5a. The lattice nodes on the common node row are present on both parts of the grid.



(a) Hanging nodes (2D) (b) Hanging nodes (3D)

Figure 3.5: Typical non-uniform grid interface in 2D and 3D

For the scaling of information between neighboring coarse and fine grids, basically two different scaling approaches exist.

In a diffusive scaling, the Mach number is reduced simultaneously with the grid spacing. Apart from grid resolution Δx it is the second important parameter controlling the convergence to the incompressible Navier-Stokes equations. As consequence, the viscosity and relaxation times are constant in the whole system. The diffusive scaling requires four time steps on the fine grid during one time step on the coarse mesh for a space refinement factor of two. More information can be found in Rheinländer [135]. In contrast to that, in the acoustic scaling , the speed of sound and the Mach number are kept constant in all subdomains. This leads to level-dependent relaxation times. Secondly, a bisection of grid spacing leads to a subcycling of 2 in time, too, as $c = const. = 1$ and $\delta t = \delta x / c$ respectively. In our implementation, the acoustic scaling is used.

At the grid interface, the conservation of mass, momentum and the continuity of the stress tensor

is crucial in order to obtain a smooth grid transition without any impact on the underlying physics. In LBGK models, the equilibrium part of the particle distribution functions is independent on the current grid level. It is a function of ρ and v, which is not affected by the scaling, so that only the non-equilibrium parts $f_i^{neq} = f_{i,c} - f_i^{eq}$ have to be rescaled:

$$f_{i,f} = f_i^{eq} + s_{cf}\left(f_{i,c} - f_i^{eq}\right) \quad \text{and} \quad f_{i,c} = f_i^{eq} + s_{fc}\left(f_{i,f} - f_i^{eq}\right). \tag{3.36}$$

s_{cf} and s_{fc} are factors resulting from the scaling of stress tensor $S_{\alpha\beta}$. It is obtained locally via

$$S_{\alpha\beta} = \nu\rho(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha}) = \left(1 - \frac{\Delta t}{2\tau}\right)\sum_i \mathbf{e}_{i\alpha}\mathbf{e}_{i\beta}\left(f_i - f_i^{eq}\right). \tag{3.37}$$

Demanding its continuity at the grid interface yields

$$\left(1 - \frac{\Delta t_c}{2\tau_c}\right)\cdot f_{i,c}^{neq} = \left(1 - \frac{\Delta t_f}{2\tau_f}\right)\cdot f_{i,f}^{neq} \tag{3.38}$$

and the following expressions for the scale factors

$$s_{cf} = \frac{6\nu + \Delta t_f}{6\nu + \Delta t_c} \quad \text{and} \quad s_{fc} = \frac{6\nu + \Delta t_c}{6\nu + \Delta t_f} \tag{3.39}$$

with viscosity ν, and time steps on the coarse Δt_c and fine grid Δt_f. In the MRT model, in analogy to the SRT approach, the equilibrium moments are independent on the grid level too. Hence, only the non-equilibrium moments m_i^{neq} have to be rescaled according to

$$m_{i,f}^{neq} = s_{cf}^i\, m_{i,c}^{neq} \quad \text{and} \quad m_{i,c}^{neq} = s_{fc}^i\, m_{i,f}^{neq} \tag{3.40}$$

with the moment-specific scale factors

$$s_{cf}^i = \frac{s_c^i}{s_f^i}\frac{\Delta t_f}{\Delta t_c} \quad \text{and} \quad s_{fc}^i = \frac{s_f^i}{s_c^i}\frac{\Delta t_c}{\Delta t_f}. \tag{3.41}$$

Δt_c and Δt_f denote the time steps on the two grid levels, whereas s_c^i and s_f^i are the collision factors (Eq. 3.18). Hence, the conserved moments for density and momentum are not rescaled at the grid interface. The resulting scale factors for the moments $m_{9,11,13,14,15}$ correspond to the scale factors for the LBGK model (Eq. 3.39).

On top of rescaling of particle distribution functions, interpolation in space and time is mandatory at the grid transition. Hanging nodes, which are present on the fine grid, are missing on the coarse grid (Fig. 3.5). To reconstruct the particle distribution functions at hanging nodes, a cubic interpolation according to Crouse [21] is used. For the time interpolation, a linear approach is used. Details on the interpolation schemes, the grid refinement and its implementation in VirtualFluids can be found in the work of Freudiger [42] and Geller [49].

Part II

Basic interface models

CHAPTER 4

LBM for free surface flows

The simulation of free surface flow problems requires the introduction of additional physics, as introduced for macroscopic methods in section 2.2. The Lattice Boltzmann method can be extended or adapted to be able to cope with free surface and multiphase flow simulations of varying complexity and space dimensions. By nature, the extensions on a mesoscopic level are somewhat different from the classical macroscopic extensions to multiphysics solvers. The mesoscopic collision operator has to be modified, and the kinematic free surface boundary condition will be fulfilled in a different way eventually. In the following, two well-known LB extensions for shallow water and free-surface flow will be presented. They serve as a basis for the GPU implementation later in this work.

4.1 LBM for shallow water equations

In chapter 3, the Lattice Boltzmann method has been derived and the relation to the incompressible Navier-Stokes equations was shown. It is self-evident, that similar mesoscopic approaches might be used for the simulation of shallow water flows. Basically, similar approaches as during the common derivation of shallow water equations (SWE) are taken. The fluid pressure is related to the water height using a different equation of state. A very comprehensive overview over the Lattice Boltzmann model for shallow water equations is given by Zhou [184], including elaborate discussions of the treatment of bottom elevation and forcing terms.

Although the LB method is not widely known in the free surface community, several groups already applied LB models to standard shallow water benchmark problems and test cases. Frandsen [39] uses a D2Q9 implementation for the simulation of wave runup on a sloping beach. Thömmes et al. [157] apply a similar Lattice Boltzmann model for test cases including bed slope and bed friction terms. The main focus of their work is to demonstrate the ability of the LB method to cope with complex geometries and irregular bathymetry, which is underlined by the simulation of the mean flow in the Strait of Gibraltar. These models are based on SRT collision operators. Tubbs [160] uses an extended

LB model with TRT and MRT collision operators. Moreover, the author extends the standard LB by a multilayer approach to consider vertical, three-dimensional effects. The model is capable of simulating wind- and density-driven circulations over irregular bathymetry.

4.1.1 Details of the Lattice Boltzmann Model

In the following, the resulting Lattice Boltzmann model for shallow water equations is briefly reviewed. The full derivation from continuous models and the Chapman-Enskog expansion to show the recovery of height-averaged Navier-Stokes equations will not be shown. For more information on these topics, the reader is referred to [184]. The Lattice Boltzmann equation for SWE with an LBK collision operator reads

$$f_i(t+\Delta t, \mathbf{x}+\mathbf{e}_i\Delta t) - f_i(t,\mathbf{x}) = -\frac{1}{\tau}(f_i - f_i^{eg}) \tag{4.1}$$

with a microscopic reference velocity $c = \frac{\Delta x}{\Delta t}$. The relaxation time τ is linked to the kinematic viscosity via

$$\nu = \frac{c^2\Delta t}{6}(2\tau - 1). \tag{4.2}$$

The relation to the classical LBE is obvious. The main differences are the definition of the reference velocity c, which is not set to unity, and the definition of the local equilibrium distribution functions. For shallow water models on the basis of D2Q9 velocity models they read

$$f_i^{eq} = \begin{cases} h - \frac{5gh^2}{6c^2} - \frac{2h}{3c^2}\mathbf{v}^2, & i = 0 \\ \frac{gh^2}{6c^2} + \frac{h}{3c^2}\mathbf{e}_i\mathbf{v} + \frac{h}{2c^4}(\mathbf{e}_i\mathbf{v})^2 - \frac{h}{6c^2}\mathbf{v}^2, & i = E,W,N,S \\ \frac{gh^2}{24c^2} + \frac{h}{12c^2}\mathbf{e}_i\mathbf{v} + \frac{h}{8c^4}(\mathbf{e}_i\mathbf{v})^2 - \frac{h}{24c^2}\mathbf{v}^2, & i = NE,SW,SE,NW. \end{cases} \tag{4.3}$$

The macroscopic values for water depth h and velocity $\mathbf{v}$ are low order moments of the distribution functions:

$$\sum_{i=0}^{b} f_i = h \qquad \sum_{i=0}^{b} \mathbf{e}_i f_i = h\mathbf{v} \tag{4.4}$$

Again, the relation to the Navier-Stokes model can be seen, whereas the density ρ has been replaced by the water height h. Note that the choice of macroscopic reference velocity c directly influences the magnitude of the discrete velocity vectors $\mathbf{e}_i$, in accordance to Eq. 3.4.

4.1.2 Body forces

In shallow water models, the body force term serves to model bed stress, wind-induced shear stress or sea bed inclination (see Eq. 2.12). To recall, a simplified force term resulting from a bottom boundary inclination reads

$$\mathbf{F} = -gh\nabla\mathbf{z} \tag{4.5}$$

with bottom coordinate z. Zhou [184] discusses various approaches for the evaluation of the forcing term in detail and finally proposes a centered scheme, where the term is evaluated at the mid-point between two neighboring lattice nodes:

$$\begin{aligned} \mathbf{F}_i &= \mathbf{F}_i\left(\mathbf{x} + \tfrac{1}{2}\mathbf{e}_i\Delta t, t + \tfrac{1}{2}\Delta t\right) \\ &= -gh\left(\mathbf{x} + \tfrac{1}{2}\mathbf{e}_i\Delta t, t + \tfrac{1}{2}\Delta t\right)\nabla\mathbf{z}\left(\mathbf{x} + \tfrac{1}{2}\mathbf{e}_i\Delta t, t + \tfrac{1}{2}\Delta t\right). \end{aligned} \tag{4.6}$$

The modified Lattice Boltzmann equation reads

$$f_i(t + \Delta t, \mathbf{x} + \mathbf{e}_i \Delta t) - f_i(t, \mathbf{x}) = \Omega_i + \Delta t F_i \tag{4.7}$$

where F_i is a link-specific contribution of the total body force $\mathbf{F}$. In the discrete, centered version, the link-specific value for F_i is evaluated as

$$F_i = \mathbf{e}_i \frac{\Delta t}{6c^2} \mathbf{F}_i. \tag{4.8}$$

Introducing the arithmetic mean for h and a central finite difference scheme for the bottom elevation derivative $\nabla \mathbf{z}$, we end up with the following expression to evaluate the force term:

$$F_i = \mathbf{e}_i \frac{\Delta t}{6c^2} (-g) \frac{h(\mathbf{x}) + h(\mathbf{x} + \mathbf{e}_i \Delta t)}{2} \frac{z(\mathbf{x}) - z(\mathbf{x} + \mathbf{e}_i \Delta t)}{\Delta x_i} \tag{4.9}$$

where h and z are water depth, or bottom boundary height respectively, of the corresponding lattice nodes. Utilizing $\Delta x = \mathbf{e}_i \Delta t$, Eq. 4.9 yields

$$F_i = -\frac{g}{12c^2} (h(\mathbf{x}) + h(\mathbf{x} + \mathbf{e}_i \Delta t))(z(\mathbf{x}) - z(\mathbf{x} + \mathbf{e}_i \Delta t)). \tag{4.10}$$

4.1.3 Boundary conditions

Similar to the LB solver for the bulk flow (section 3.4) the boundary conditions in the LB context have to be specified for the particle distribution functions directly and can not be specified by solely prescribing macroscopic quantities for the primary variables water height and depth-averaged velocity.

No-flow boundary conditions

For no flow boundary conditions, a simple bounce back scheme may be used. In analogy to the fully three dimensional flow kernel, particles bounce off the boundaries and are reflected. This way, no slip and slip boundary conditions can be modeled. In both cases, the velocity perpendicular to the wall and hence the flow through the wall equals zero.

Moving wall boundary conditions

For moving boundaries, a modified bounce-back scheme [101] serves to adjust the resulting fluid velocity. A velocity-dependent term is added to the particle distribution functions in every time step:

$$f_i = f_I + \frac{2\,\rho}{c_s^2} \omega_i \mathbf{e}_i \mathbf{v}. \tag{4.11}$$

Note that this boundary conditions also leads to a zero-lateral velocity, as it is a modified no slip boundary condition.

Water height and flux boundary conditions

For non-trivial Dirichlet and Neumann boundary conditions, the modified bounce-back scheme for moving boundaries (Eq. 4.11) may be used. This scheme also is proposed by Zhou [184], although his notation differs. Evaluating the terms of Eq. 4.11 leads to the following simplified expressions for a west boundary

$$f_E = f_W + \frac{2 \cdot q_x}{3c}, \tag{4.12}$$

$$f_{NE} = f_{SW} + \frac{q_x}{6c} + \frac{f_S - f_N}{2}, \tag{4.13}$$

$$f_{SE} = f_{NW} + \frac{q_x}{6c} + \frac{f_N - f_S}{2}, \tag{4.14}$$

where the eastward-pointing distribution functions are missing. It can be seen that, in the end, only the term for the boundary flux q_x shows up in the equations. Hence, if a water height $\bar{h}$ is specified instead of a flux $\bar{q}_x$, it can be converted to an equivalent flux: $\tilde{q}_x = \bar{h}\, v_x$. In the literature, this formulation is used for all kinds of boundary conditions, no matter if velocity, flux or water height is given (see Thömmes et al. [157]). In our test cases, it has shown to lead to oscillations, as a product of two factors is used for the flux calculation, and at least one factor (v_x) has either to be taken from the previous time step or extrapolated from a neighboring grid location. We propose an alternative formulation for the water height boundary, referring to the anti-bounce back pressure boundary condition for the LB fluid solver (Eq. 3.23):

$$f_I^{t+1} = -f_i^t + f_I^{eq}(h_B, \mathbf{v}) + f_i^{eq}(h_B, \mathbf{v}). \tag{4.15}$$

with Maxwellian equilibrium distribution functions $f_{i,I}^{eq}$.

4.1.4 Stability requirements

Nearly all explicit numerical methods suffer from more or less severe stability requirements. The stability requirements of the shallow water model are less straightforward to discover. Zhou [184] identifies the most severe requirements to be

$$\mathrm{CFL} = \frac{|\mathbf{v}|}{|\mathbf{c}|} \leq 1.0 \quad \text{and} \quad \mathrm{Fr} = \frac{|\mathbf{v}|}{\sqrt{gh}} \leq 1.0 \tag{4.16}$$

$$\nu = \frac{c^2 \Delta t}{6}(2\tau - 1) > 0 \quad \text{and} \quad \frac{\mathrm{CFL}}{\mathrm{Fr}} = \frac{\sqrt{gh}}{|\mathbf{c}|} \leq 1.0 \tag{4.17}$$

The CFL number, Froude number and their ratio has to be smaller than one, and negative viscosity is not permitted. Typically, the space discretization is set first, fixing Δx. Secondly, by selecting a proper timestep Δt the macroscopic reference velocity $c = \Delta x / \Delta t$ is defined.

4.1.5 Shoreline algorithm

Solutions of the shallow water equations are only valid for a water level h greater than or equal to zero. Hence, the treatment of drying or wetting boundaries (as in wave runup simulations) requires

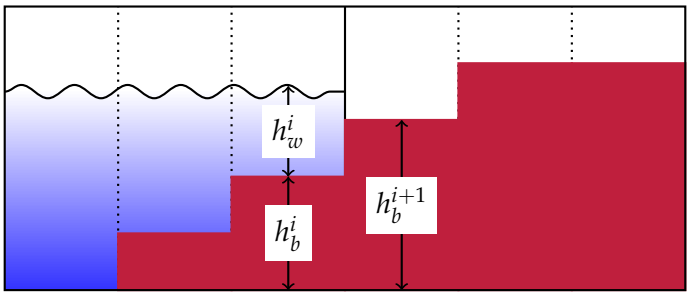


Figure 4.1: A simple shoreline algorithm

additional consideration. Shoreline algorithms are used to artificially adjust the domain size in the runup region, depending on the rising or falling water level at the boundary nodes.

We use a simple wetting and drying model, where the water level in the vicinity of the solid boundaries is checked via evaluating

$$h_b^i + h_w^i > h_b^{i+1} + \delta \tag{4.18}$$

with a proper threshold value δ, see also Fig. 4.1. If the water is overtopping, the neighboring inactive solid node is activated and converted to a fluid node. The equilibrium distribution functions for an extrapolated water height and zero velocity are applied:

$$f_i = f^{eq}\left(h, \mathbf{0}\right) \forall i. \tag{4.19}$$

For test cases with a characteristic wave propagation direction which is aligned to the coordinate axes, this approach leads to accurate results. For more complex bottom topographies, advanced runup models have to be used. Elaborate discussions of ten different methods, mainly for implicit methods, and a presentation of demanding runup test cases can be found in the work of Balzano [8]. Lynett et al. [109] proposes an extrapolation technique near the wet-dry interface to determine artificial fluid properties in the dry region. For the test cases used in this work, the simple staircase-shaped method works fine.

4.2 LBM for three-dimensional free surface flow

Several approaches have been developed to use the LBM for free surface flow simulations. Gunstensen et al. [68] propose the immiscible Lattice Boltzmann (ILB) model, a multiphase model combined with an additional anti-diffusion sweep (recoloring step) which prevents the mixing of the two phases. Ginzburg and Steiner [55] modify this approach and neglect the second fluid phase. In contrast to the underlying multiphase model, the LB calculation steps are only performed on the nodes of the wet phase. Lallemand et al. [103] combine an Eulerian LBM for the flow field and a Lagrangian front-tracking method for the advection step. In this hybrid method, the interface is captured through marker particles, which are then advected on the basis of a valid pressure and velocity field. We recently presented a hybrid scheme that discretizes the advection equation in a classical macroscopic way on the basis of valid pressure and velocity fields provided by the LBM, see Janßen and Krafczyk [90]. This will be discussed in detail in Part III of this thesis.

Apart from the aforementioned methods, Körner et al. [95] and Thürey and Rüde [156] combine LBM with a VOF method and a flux-based advection scheme. Their algorithm was developed initially for the simulation of metal foams, but is capable of handling free-surface flow simulations as well. Opposite to common VOF methods, the flux terms are expressed directly in terms of LBM distribution functions. The straightforward surface reconstruction and the time-explicit advection made us select this free surface capturing scheme as the basis of our GPU implementation.

4.2.1 Details of the free surface Lattice Boltzmann Model

The approach is based on a VOF interface capturing method. In a VOF method, the interface is caputured via the *fill level* ε of a cell, which qualifies the amount of a cell which is filled with fluid:

$$\varepsilon = \frac{V_{fluid}}{V_{cell}}. \tag{4.20}$$

A fill level of 0.0 marks an empty cell in the inactive gas domain, a fill level of 1.0 corresponds to a filled cell inside the fluid domain. Fluid and gas cells are separated by a closed interface layer (Fig. 4.2a) with a fill level between 0.0 and 1.0. In our LB context, we assign one VOF control volume to one lattice node. In order to calculate the evolution of the free surface in time, an additional advection equation has to be solved. The flux between two cells can be evaluated in terms of particle distribution functions f:

$$\Delta m_i = \left[f_I\left(\mathbf{x}, t\right) - f_i\left(\mathbf{x}, t\right)\right] \cdot A_i \tag{4.21}$$

where I is the inverse direction to i. A_i denotes the wet area between two cells and can be estimated as arithmetic mean of the fill level of two neighboring cells:

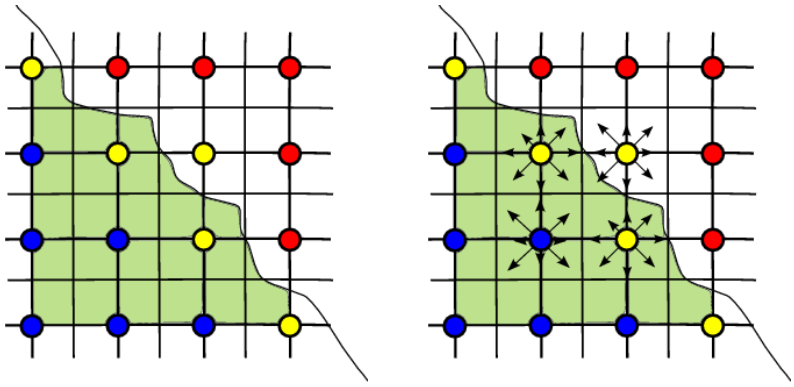
$$A_i = \begin{cases} 1.0 & \text{: neighbor FLUID cell} \\ \frac{\varepsilon(\mathbf{x},t)+\varepsilon(\mathbf{x}+\mathbf{e}_i,t)}{2} & \text{: neighbor INTERFACE cell} \\ 0.0 & \text{: neighbor GAS or SOLID cell} \end{cases} \tag{4.22}$$

Opposite to higher-order schemes, the normal vector information is not considered. Hence, this approach does not reprocude all line (2D) or plane (3D) segments exactly and is of first order in

space. Once the flux terms have been evaluated, the new fill level of a cell can be calculated via

$$\varepsilon^{n+1} = \frac{\rho^n \varepsilon^n + \sum_i \Delta m_i}{\rho^{n+1}} \tag{4.23}$$

where $\rho^{n,n+1}$ is the fluid density at time step n or $n+1$, respectively, (Eq. 3.8) and ε^n is the fill level at time step n [95, 156].



(a) Fluid, interface and gas nodes (b) Flux between interface nodes

Figure 4.2: Fluid (blue), interface (yellow) and gas (red) nodes

4.2.2 Resulting algorithm for the node update

After the fill levels ε have been updated, the consistency of node state and new fill level has to be assured. In the advection step, cells with a fill level larger than 1.0 or lower than 0.0 can appear, as the interface evolves in time. Consequently, draining cells change their state from interface to gas, and neighboring fluid nodes have to become interface nodes. Analogously, the cells which have been filled up change their state from interface to fluid. The neighboring gas nodes, which were inactive before, have to become interface nodes too. These new interface nodes have to be initialized, as they do not contain any valid distribution functions. Therefore the macroscopic values of density and velocity (Eq. 3.8) from neighboring existing fluid nodes are interpolated:

$$\bar{\rho}(\mathbf{x}) = \sum_i w_i \rho(\mathbf{x} + \mathbf{e}_i) \quad \text{and} \quad \bar{\mathbf{v}}(\mathbf{x}) = \sum_i w_i \mathbf{v}(\mathbf{x} + \mathbf{e}_i). \tag{4.24}$$

Based on these information the particle distribution functions f are initialized with Maxwellian equilibrium distribution functions:

$$f_i = f_i^{eq}(\bar{\rho}, \bar{\mathbf{v}}). \tag{4.25}$$

The equilibrium distribution functions modified for incompressible flows are given in Eq. 3.14. A local, LB-specific, Poisson-type iteration [114] might be used for the improvement of the non-equilibrium part of the distribution functions. The resulting overall algorithm is given in Alg. 4.1.

Algorithm 4.1: Calculation loop for one time step and one grid node in a basic LBM free surface scheme

```
// Basic LB scheme;
if node type == fluid then
    collision;
    forcing;
    propagation;
    boundary conditions;
end
// Update the interface;
if cell type == interface then
    determine wet area for all lattice directions;
    calculate mass flux and evaluate new fill level;
    if new fill level ε_i^{t+1} < 0.0 then
        convert cell to gas cell;
        remember cell coordinate for the closing consistency check;
    end
    if new fill level ε_i^{t+1} > 1.0 then
        convert cell to fluid cell;
        remember cell coordinate for the closing consistency check;
    end
end
check and - if necessary - convert neighbor lattice nodes of cells that changed state;
initialize new interface nodes;
```

CHAPTER 5

Enhanced LB Simulations on GPUs

Characteristic free-surface flow applications require fast, three-dimensional, turbulent and highly resolved simulations, so that the demand for powerful simulation frameworks is larger than ever before. Even hybrid models, which combine fluid models of varying complexity to minimize simulation times, cannot solve this problem alone. We recently coupled a three-dimensional VOF solver to a two-dimensional potential flow code, in order to save computational time in regions of low interest, e.g. far away from wave breaking. Nonetheless, the run-times of the three-dimensional solver drastically slow down the coupled simulation ([89],chapter 8). Hence, in order to be able to compute large computational domains in a reasonable amount of time, the utilization of parallel hardware is crucial. GPGPUs (General Purpose Graphics Processing Units) recently introduced high-performance computing to the desktop PC, to run large-scale simulations locally without the tedious access and data transfer to high-performance computers. Recently released software development kits (SDKs) such as the new nVIDIA CUDA technology [126, 93, 140] allow computationally intensive applications to access the processing power of a GPU.

This chapter deals with the implementation of the previously described Lattice Boltzmann schemes for shallow water equation and fully non-linear three-dimensional free surface flow for numerical simulations running on GPGPUs. Both, the basic implementation and the multiphysics extensions are validated with reference data and are applied to state-of-the-art bulk flow and free surface test cases.

5.1 State of the art

In the recent years, GPUs have matured from toys for gamers to serious high-performance computing hardware. Apart from classical CFD, GPUs also play an important role in computational steering. The main idea of a computational steering environment is to provide a tool for engineers to design and check the quality of designs, desirably in real time. GPUs have rendered this possible, at least in

conjunction with a well-chosen numerical method. Linxweiler et al. [107] lately presented a computational steering environment for CFD, which is based on a LBM implementation on GPU architecture. It allows the user to interactively place and move objects in a turbulent flow and to immediately see the influences on the flow patterns and vorticity. A competitive GPU implementation of a free surface algorithm could easily be included in this computational steering environment. Apart from supercomputing on the desktop, GPU clusters combine massively parallel CPU cores with high GPU power, such as the recently installed HPC cluster LUDWIG (TU Braunschweig), which extends 1400 CPU cores with 96 nVIDIA Tesla c1060 GPUs, leading to an overall theoretical peak performance of more than 100 TFlops.

Several authors accelerated their LBM computations on general-purpose graphics hardware, also concerning multiphysics and even before the graphics vendors started to develop their software development kits. LBM usually operates on a finite difference grid, is explicit in time and usually requires only next neighbor interaction. Hence, it is very suitable for the implementation on GPUs. The applications range from simulation of soap bubbles (Wei et al. [170]) to the simulation of miscible binary mixtures (Zhu et al. [185]) and melting and flowing in multiphase environment (Zhao and et al. [183]). Recently, GPU clusters have been assembled for general-purpose computations (Fan et al. [35]) and LB simulations have been performed. With the development of SDKs, the programming style is not as close to the hardware as before. Tölke and Krafczyk [158, 159] implemented two-dimensional and three-dimensional LB models on nVIDIA GPUs and could show an efficiency gain of more than one order of magnitude compared to a CPU code.

Simulations of long wave propagation on the basis of shallow water equations are very demanding in terms of hardware and grid resolution. A typical wave-propagation problem involves effects on an overall length scale of hundreds of kilometers. If a grid resolution in the range of meters is desired (and it is, for sufficient accuracy), high computational performance is needed. Recently, Geveler et al. [52] presented a Lattice-Boltzmann approach for the simulation of shallow water equations, targetting various architectures, including GPUs. Tubbs [160] lately presented a shallow water kernel utilizing a MATLAB toolbox for enabling computations on nVIDIA GPU hardware. Also in the field of three-dimensional free surface flow, GPUs have been successfully applied, especially using smoothed particle hydrodynamics (SPH), see. e.g. [182] and [71]. Recently, a GPU version of the open source SPH free surface flows code SPHysics has been developed and is maintained and used by several research groups, see Dalrymple and Herault [23] and Herault et al. [79].

5.2 GPU Implementation of the LB bulk scheme

For the implementation of the LBM algorithm on GPU hardware, we use the nVIDIA CUDA Toolkit, which is an entire software development solution for programming CUDA-enabled nVIDIA GPUs. The toolkit gives computationally intensive applications access to the processing power of a GPU. Our simulations are carried out on nVidia hardware only, namely on the nVIDIA GTX 275 and Tesla C1060. The former card provides a device memory of 1 GB and 240 cores at 1.4 GHz clock speed each. For our enriched D3Q19 free surface model, 165 Bytes per lattice node have to be allocated, which yields a maximum number of 6.5E6 lattice nodes. The latest generation of Tesla GPUs (Tesla C1060, as installed in Ludwig) provides up to 4 GB of device memory, which quadruplicates the

maximum number of lattice nodes. In the following, the mapping of the lattice nodes to the parallel architecture is shown and details on the implementation of boundary conditions are given.

5.2.1 Topology

The GPU is a computing device with a large amount of cores, each executing a number of threads. To arrange these threads, the CUDA toolkit offers a two-level parallelism (see Fig. 5.1). First, all threads are grouped in one *thread block*. In a thread block, extremely fast shared memory is available among the threads and the threads can be synchronized. Each thread is identified by its three-dimensional thread index, which is the position in the thread block. To utilize the hardware efficiently, the total number of threads per block should be in the range of 64 to 512. The maximum number of threads can be smaller due to the limited amount of local and shared memory which is available on the particular GPU. Secondly, the thread blocks are bundled in the *grid*. Opposite to threads in one and the same block, threads in different blocks can only communicate via the device memory and a syncronisation is not possible. Blocks are identified by their two-dimensional block index, namely the position in the grid. Further details on the thread processing, the grouping in warps and the distribution among the GPU multiprocessors can be found in [127].

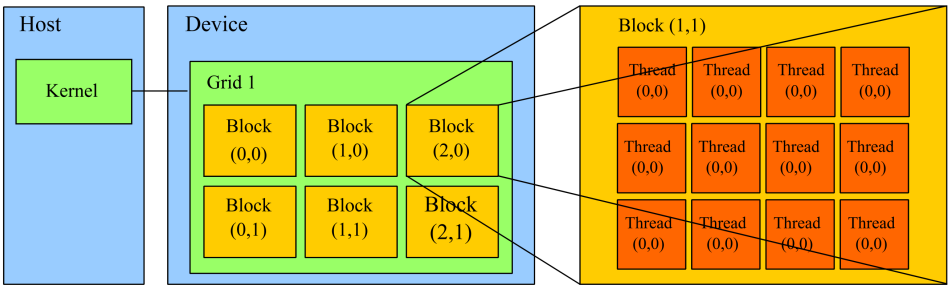


Figure 5.1: Two-level parallelism of the nVIDIA CUDA SDK [127]

5.2.2 Grid mapping

The main design element in the GPU implementation of a numerical method is the mapping of the grid to the computational hardware, i.e. in our case the mapping of lattice nodes to the grid, blocks and threads. Several restrictions for the memory access pattern have to be considered in order to achieve maximum performance. In particular, the thread k must access the k-th word in a memory segment aligned to 16 times `sizeof(float)`. When these access requirements are met, global memory accesses are coalesced by the device in one single transaction. If this pattern is violated, the memory accesses can not be coalesced and the performance drops remarkably. Although the recently released graphics cards with a compute cabability larger than 1.1 offer a higher flexibility, we decided to keep the below-mentioned optimized memory access pattern, not least for the sake of backward compatibility of the code. Habich et al. [70] recently presented elaborate discusssions of D3Q19 performance, using more sophisticated memory access strategies on modern GPUs.

In our grid mapping, we assign one single lattice node to one CUDA thread, which is demonstrated for the D3Q13 model in Fig. 5.2. The memory is allocated as one-dimensional array, and the memory

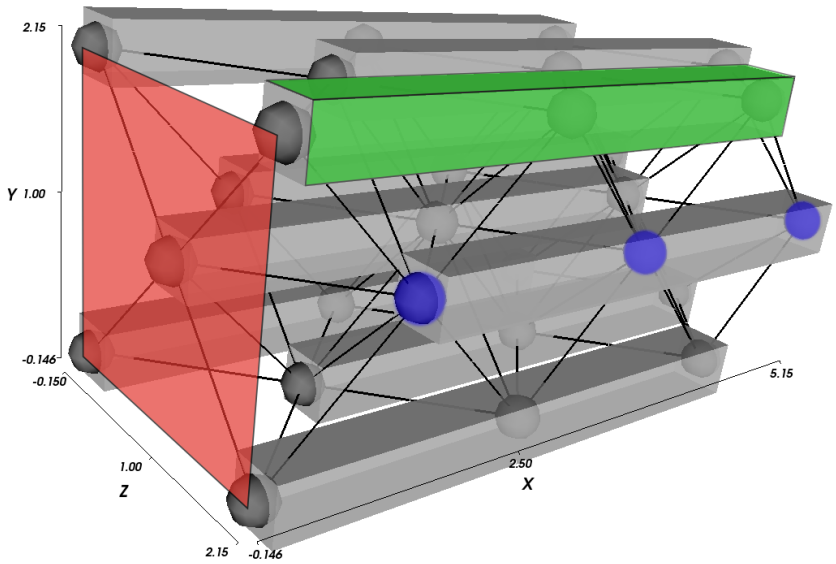


Figure 5.2: CUDA grid mapping for a D3Q13 LB model [159]: grid (red), block (green), threads (blue)

index is calculated via `k = nx*(ny*z+y)+x` for a node at position $\mathbf{x} = (x, y, z)$ and a total of nx x ny x nz nodes. Consequently, the PDFs propagating in x direction, are copied to a memory position with a `sizeof(float) = 4 Byte` shift. Due to the above-mentioned restrictions, this access to neighboring memory positions is very slow. To solve this problem, Tölke and Krafczyk [159] proposed to propagate those particle distribution functions via the shared memory which is available in one thread block. Consequently, all nodes along one line in x direction have to be gathered in one single thread block. The y and z dimensions are mapped to the grid, so that the coordinates of a node can be determined via `x = threadId.x`, `y = blockIdx.x` and `z = blockIdx.y`

The total amount of shared memory is currently limited to 16kB. In the D3Q19 model, ten PDFs propagate in positive or negative x direction (Eq. 3.3), so that 40 Bytes of shared memory per lattice node are needed, which corresponds to a maximum number of 400 nodes in one thread block. If the x-dimension of the computational domain exceeds this maximum number of threads, the domain also has to be partitioned in x-direction The CUDA *grid*, which groups the thread blocks, is limited to two dimensions and can not deal with two consecutively aligned thread blocks in a third direction. Hence, this feature is included in the kernels explicitly. The block index in x direction (`blockIndexX`) is passed to the kernels and the new x-coordinate is calculated manually as `x = threadId.x + blockIndexX*nodes_per_threadBlock`.

The dimension of the grid and the number of threads is passed to the kernel, and CUDA manages the exact distribution of tasks among the multiprocessors and cores. A common collision and propagation kernel bundles the collision and propagation steps (Eq. 3.5) in order to reduce additional memory accesses. The distribution functions are fetched from the main device memory, the collision takes place, and the post-collision distribution functions are written to the neighboring memory positions. For the distribution functions propagating in x direction, an intermediate memory transaction in the shared memory is used, as depicted in Fig. 5.3.

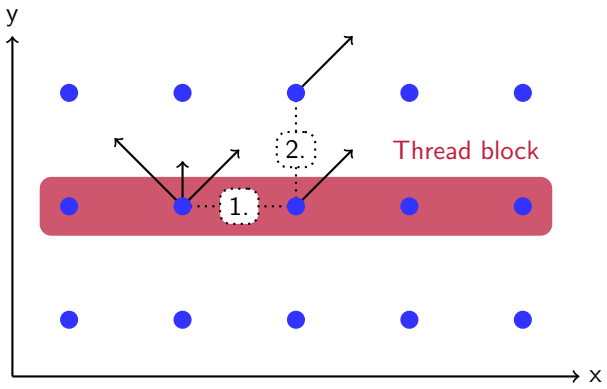


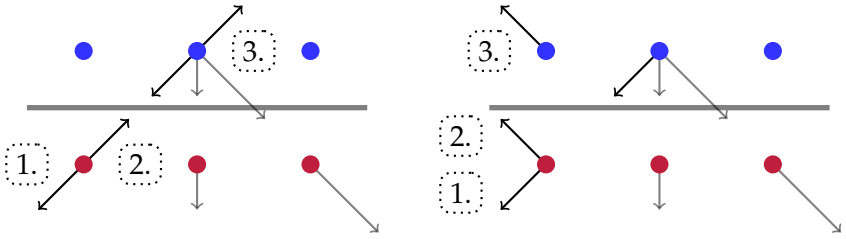
Figure 5.3: CUDA grid mapping and LB propagation via the shared memory of a thread block

5.2.3 Boundary conditions

From the algorithmic point of view, boundary conditions disturb the original, homogeneous algorithm as they require additional operations on only a certain number of nodes. In general, a unique LB kernel for all lattice nodes is preferrable on a data- and thread-parallel system for a homogeneous load balancing.

The no-slip and velocity boundary conditions can be incorporated in the LB collision and propagation kernel, as the global memory access pattern for the propagation remains the same as for the classic propagation step. Instead of the collision step, the boundary nodes simply reflect the incoming particle distribution function and - if neccessary - add the contribution of the boundary velocity $\bar{\mathbf{u}}$. After that, the unmodified propagation step takes place, which advects the reflected distribution functions to the corresponding neighbor, as shown in Fig. 5.4a. Ghost layers surround the whole computational domain, so that all eighteen particle distribution functions can safely be advected to the neighboring nodes, even at the domain boundary.

Opposite to that, the bounce-forward scheme for slip boundary conditions violates the common propagation pattern. The distribution functions are not reflected to the donating lattice node but bounce forward to the neighboring lattice node. Consequently, the BC does not fit into the common propagation pattern and requires an additional kernel for the advection. The solid nodes at the boundary receive particle distribution functions from neighboring fluid nodes during propagation. Subsequently, a second kernel is executed and shifts the particle distribution functions to the correct location, see Fig. 5.4b



(a) No-slip (b) Slip

Figure 5.4: GPU implementation of implicit no-slip and slip boundary conditions

The same holds for the extrapolation boundary conditions. A separate kernel performs a copy operation, where the complete set of distribution functions of the next to last fluid node is copied to the last one.

As the free surface location varies in time, this boundary condition can only be partially optimized. The free surface boundary condition on interface nodes additionally needs to read the node state at neighboring memory locations. If - and only if - the neighboring node is of gas state, the pressure boundary condition has to be applied, which leads to non-local memory read operations.

5.2.4 Force evaluation

The evaluation of the forces on obstacles in the flow (Eq. 3.27) requires a loop over the entire set of lattice nodes, summing up the nodal contribution to the force vector. For performance reasons, and since a continuous force evaluation in every time step is preferrable, the force is evaluated directly on the GPU. All-reduce operations in thread-parallel systems always demand for a careful treatment to avoid race conditions among the threads. A first kernel computes the contribution of one thread block to the total force. Each single node evaluates Eq. 3.27, and the first thread in the whole thread block is responsible for the summation: $\tilde{F}_B = \sum_{nx} F_i$. A thread-global synchronisation point guarantees that all threads finish their force calculation before the summation. The resulting force of one thread block is stored in the global device memory, resulting in a matrix of `ny` times `nz` entries. In the following, these sub-forces are accumulated by two additional loops: $\mathbf{F} = \sum_{ny} \sum_{nz} \tilde{F}_B$.

5.2.5 Validation of the flow solver

In this section, the implementation of the D3Q19 bulk flow scheme is validated and analyzed. This way, the maximum performance of the CUDA implementation without additional multiphysics extensions is obtained. In a second step, the shallow water and free surface extensions are activated and the impact on the overall performance can be analyzed. For the comparison of perfomance data, *MNUPS* is the most common performance measure in the Lattice Boltzmann community and corresponds to Million Node Updates Per Second.

Poiseuille flow between infinite plates

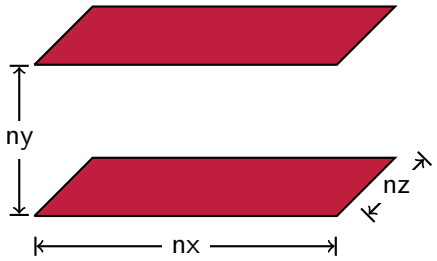
The performance and accuracy of the fluid solver is demonstrated with a Poiseuille flow between plates on several different grid configurations. In this straightforward problem, the fluid is moving laterally between two plates with infinite length and width. As the grid is refined, the viscosity and the body force are adjusted to match the fixed Reynolds and Mach numbers given in test case setup 5.1.

The analytical solution for the velocity and pressure profiles is known and can be used to validate the accuracy of the solver. The flow is driven by a pressure gradient and retarded by viscous drag along both plates. Demanding balance of these forces leads to the following solution for the flow velocity:

$$u_x(z) = \frac{z^2 - (0.5L_z)^2}{2\nu} \frac{dp}{dx} \tag{5.1}$$

Parameter	Value
Grid	see Tab. 5.1
Re	100
Ma	0.017
BC x	periodic
BC y	periodic
BC z	no-slip

(a) Parameters



(b) Geometry

Test case 5.1: Poiseuille flow between infinite plates

The resulting performance on a Tesla C1060 and the L1 error norm for the maximum velocity in the channel are given in Tab. 5.1. We can see a performance maximum of 358 MNUPS for a grid resolution of 64x96x96 nodes. The average performance yields approximately 320 MNUPS. Concerning the error norm, the expected convergence behaviour can clearly be observed. Grid refinement in flow direction (x) does not affect the quality of the result.

	nx				
ny, nz	32	64	128	256	Error norm
32x32	288	337	249	346	7.9321E-04
64x64	289	329	346	353	2.2706E-04
96x96	293	358	328	351	1.5168E-04
128x128	320	276	325	330	1.1534E-04
192x192	260	358	288	327	9.0002E-05
256x256	318	358	346	356	9.6668E-06

Table 5.1: Performance for the Poiseuille flow problem (MNUPS)

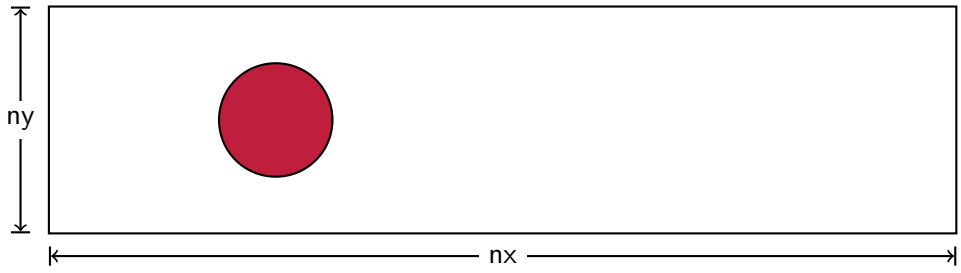
Moving sphere in a circular pipe

A second test case for the calculation of the flow field at the same time serves to verify the force evaluation method, which is needed for the force calculation in the numerical wave tank later on. We simulate a moving sphere in a pipe at a low Reynolds number on four gradually refined grids. We choose a coordinate system moving with the sphere, as shown in test case setup 5.2. The setup is axially symmetric, so that only a 2D cut is shown. No-slip conditions are imposed on the boundary of the sphere and velocity boundary conditions are imposed on the inflow, outflow and on the boundary of the pipe.

As the grid is refined, the inflow velocity and hence the Mach number is lowered to maintain a constant Reynolds number $\mathrm{Re} = 1$. The resulting drag force F_D on the sphere is evaluated by means of Eq. 3.27 and is transferred to the dimensionless drag coefficient

$$c_d = \frac{2F_D}{\rho_w \pi d^2 U^2} \tag{5.2}$$

which can be compared to reference values. An approximate solution for the dimensionless drag



Test case 5.2: Moving sphere in a pipe

coefficient for a sphere moving with speed u_0 in an infinite fluid is given by [144]:

$$c_d = \frac{24}{\mathrm{Re}}\left(1 + 0.15\mathrm{Re}^{0.687}\right) \tag{5.3}$$

where the Reynolds number is defined as

$$\mathrm{Re} = \frac{u_0 D_{sphere}}{\nu}. \tag{5.4}$$

If the sphere is moving in a finite width domain (as e.g. represented by a circular pipe of diameter D_{pipe}), additional corrections have to be made to the c_D value. These are given in [36] as a perturbation expansion of $\lambda = D_{sphere}/D_{pipe}$, yielding

$$c_d = \frac{24}{\mathrm{Re}}\left\{0.15\,\mathrm{Re}^{0.687} + \frac{1 - 0.75857\lambda^5}{1 - 2.1050\lambda + 2.0865\lambda^3 - 1.7068\lambda^5 + 0.72603\lambda^6}\right\}. \tag{5.5}$$

It predicts the drag coefficient with a 5% error for $\lambda < 0.6$ and $\mathrm{Re} < 50$. The reference value for the drag coefficient in our simulation with $\lambda = 0.5$ is $c_d \approx 144.48$. In Tab. 5.2, the results are given. One can clearly observe a convergence behaviour for c_d with increasing mesh resolution.

	64x16x16	**128x32x32**	**256x64x64**	**384x96x96**	
u_0	0.008	0.004	0.002	0.001333	
ny, nz	16	32	64	96	
D_{pipe}	14	30	62	94	
D_{sphere}	7	15	31	47	Simulation
ν	0.056	0.06	0.062	0.062667	Setup
F_x^{LB}	0.1594	0.2083	0.2195	0.2239	
C_d^{LB}	129.47	147.32	145.40	145.19	
C_d^{ref}	144.48	144.48	144.48	144.48	
Rel. error	0.1039	0.0196	0.0064	0.0049	Results

Table 5.2: Simulation setup and numerical results for the drag coefficient of a moving sphere in a pipe, Re = 1 and $\lambda = 0.5$

5.3 Shallow water kernel

After initial validations of the 3D LB bulk flow kernels, the shallow water extension is adressed. The LB shallow water model is based on a two- dimensional, nine-speed stencil (D2Q9). The CUDA implementation of the 2D LB kernels is straightforward and slightly easier than the implementation of their 3D equivalents and will not be discussed in this work. Basically, the same grid mapping techniques and shared memory propagations as for the 3D LB kernel are used. Details on the 2D implementation of an LBM bulk scheme are given in Tölke and Krafczyk [158].

5.3.1 Validation

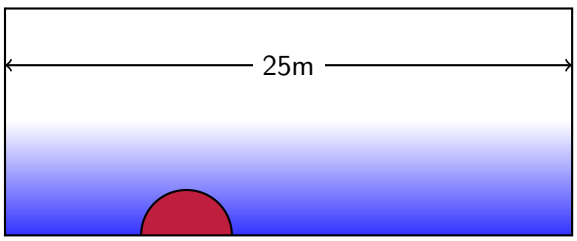
Three validation test cases serve to validate the LBM model and its implementation. After the initial simulation of the flow over a bump, where Bernoulli's equation provides an analytical reference solution for the water height, two state-of-the-art benchmark problems of the tsunami community are analyzed.

Flow over a bump

A very simple test case for the shallow water scheme including a complex bottom elevation is the flow over a bump, which also has been discussed in [184, 157]. At the left boundary, a flux q is prescribed. Above the bump, the flow velocity is increased, whereas the water level decreases, as predicted by Bernoulli's equation: as the water height contributes linearly to the overall energy while the velocity has a quadratic contribution, the water level even falls below the surrounding mean water level.

Parameter	Value
Grid	see Tab. 5.1
c	15 m/s
$1/\tau$	1.5
BC x	$q = 4.42m^2/s$
	$h = 2m$
BC y	periodic

(a) Parameters



(b) Geometry

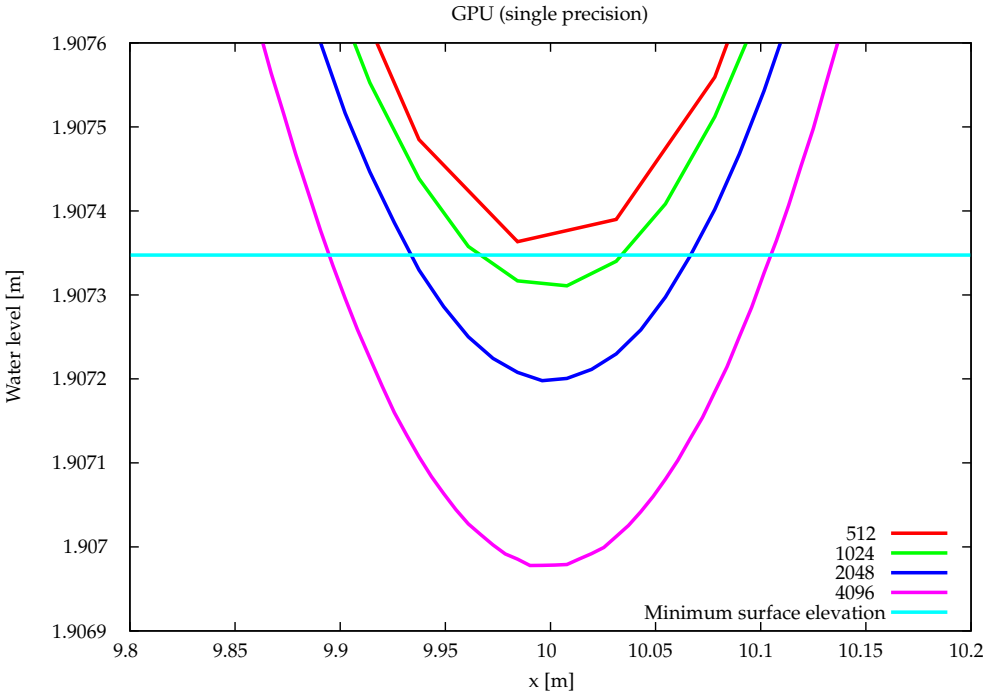
Test case 5.3: Flow over a bump

The channel is 25m long, and the bottom elevation is defined as a piecewise polynomial of the form

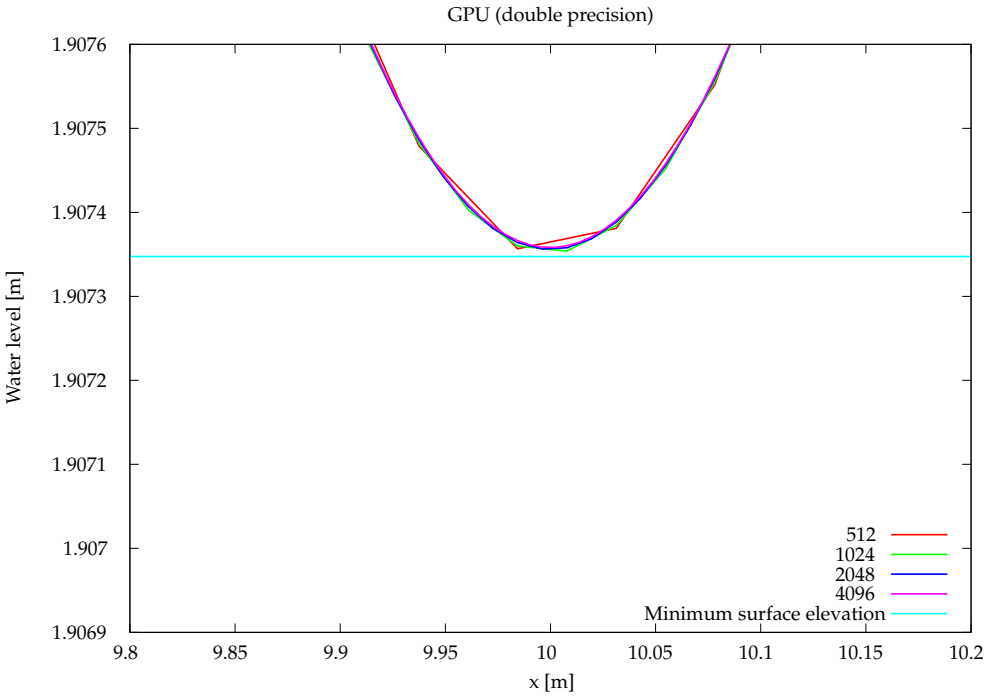
$$z_b(x) = \begin{cases} 0.2 - 0.05(x-10)^2 & 8 < x < 12 \\ 0.0 & x \leq 8, x \geq 12 \end{cases} \tag{5.6}$$

The functional $z_b(x)$ is continuous, but not continuously differentable. The kinks at $x = 8m$ and $x = 12m$ lead to a demanding test case in terms of bottom elevation forcing terms. At the inflow boundary, a constant inflow of $q = 4.42m^2/s$ is set, while the outflow boundary fixes the water level to $h = 2m$. Hence, the resulting flow velocity far away from the bump equals $v = 2.21m/s$, which also is used in the grid initalization process. Values for macroscopic reference velocity and relaxation

time are set in accordance to [184] to $c = 15m/s$ and $\tau = 1.5$. In the cross section directly above the bump at $x = 10m$, Bernoulli's equation predicts a minimum water level of $1.90735m$ at a maximum flow velocity of $2.5888m/s$. In Fig. 5.5, the numerical results for the water depth over the bump are given for several gradually refined grids and for simulations using single and double precision floating point variables. Convergence can be clearly observed, although the single precision kernel (Fig. 5.5a) shows inconsistent convergence behaviour. Additionally, the choice of macroscopic reference velocity c influences the accuracy of the single-precision results drastically. It controls the CFL number and the ratios u/c and h/c which are used in the calculation of the Maxwellian equilibrium distribution functions. If double precision variables are used (Fig. 5.5b), the results converge to the correct analytical solution. Similar effects have been observed by Schönherr et al. [145] in the GPU simulation of bulk flows. The authors compare compressible and incompressible LB formulations and diagnose a higher sensitivity of the compressible LB approach, due to the treatment of single precision float variables. The shallow-water-type LB model resembles the compressible LBGK formulation for bulk flows, as can be seen by comparing the definition of the Maxwellian equilibrium distribution functions (Eq. 3.13 vs. Eq. 3.14), so that our model apparently suffers from related effects. Elaborate discussions of the effects of single and double precision variables on the accuracy of Lattice-Boltzmann simulations can be found in Harvey et al. [76]. Nonetheless the relative error in water height h above the bump is below $1 \cdot 10^{-4}$ and hence does not have a severe impact on the quality of the overall result.



(a) Single precision



(b) Double precision

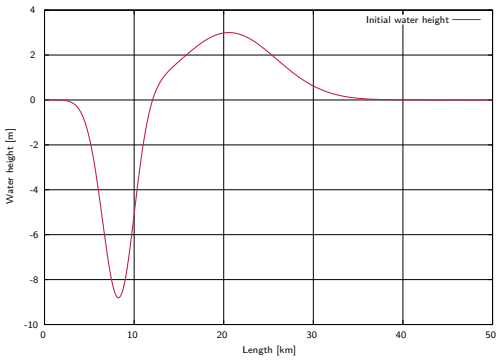
Figure 5.5: Water surface elevation above the bump for single and double precision GPU simulations in comparison to the expected minimum water level

Tsunami runup onto a plane beach

After the basic validation, two state-of-the-art test cases are adressed. The following benchmark problem of the tsunami run-up onto a plane beach was initially proposed for the "Third international workshop on long-wave runup models" which took place from June 17-18 2004 in the Wrigley Marine Science Center, Catalina Island, California. The benchmark data has been produced by the initial-value-problem (IVP) technique [16]. This and other benchmark test cases can be accessed online[1].

Parameter	Value
Domain	51.2km x 6.25m
Lattice	8192 x 4
Δx	6.25m
c	1000 m/s
Δt	6.25ms
$1/\tau$	1.95
δ	0.0001
BC x	Run-up
	No-slip
BC y	Periodic

(a) Parameters



(b) Initial wave profile

Test case 5.4: Tsunami run-up onto a plane beach

For this runup problem, the slope of the beach (1:10) and the initial wave profile (test case setup 5.4) are given. The benchmark task is to present the snapshots of the free surface in the run-up region at three time steps ($t = 160\text{s}, 175\text{s}, 220\text{s}$). Hence, the algorithm for the treatment of the air-water-beach interface at the shoreline is the key of this benchmark. Fig. 5.6 compares the resulting free surface profile in the runup region to the benchmark reference data. Very good agreement can be seen, even at relatively low resolutions. The simulation was run on a nVIDIA Tesla C1060 GPU with an average performance of 50 MNUPS, leading to an overall simulation time of approximately 30 seconds, nearly ten times faster than real time. The outstanding high GPU performance which has been observed in the previous LB bulk flow simulations can not be accessed in this benchmark because of the small amount of only four lattice nodes in y-direction. Either a wider LBM domain or more simplified, one-dimensional LB models as the D1Q3 should better be used as performance benchmarks. Frandsen [40] analyzed the same test cases by means of a D1Q3 LB model on a lattice of 100,000 nodes (resulting in a grid spacing $\Delta x = 0.5m$ and chosing a time step $\Delta t = 0.002s$). The CPU time for the whole simulation was between 6,700 and 11,000 seconds.

The differences of numerical and benchmark result on the ultimate nodes in the runup region are due to the low grid resolutions of $\Delta x = 6.25m$ resulting in a vertical bottom elevation difference of $\Delta z = 0.625m$ between two neighboring lattice nodes. For a simple runup model without extrapolation or more complex reconstruction ot the bottom elevation, this is a limiting factor. Grid refinement further improves the runup behavior, while the overall flow is well-represented even at low grid resolutions.

[1] `http://isec.nacse.org/workshop/2004_cornell/benchmark.html`

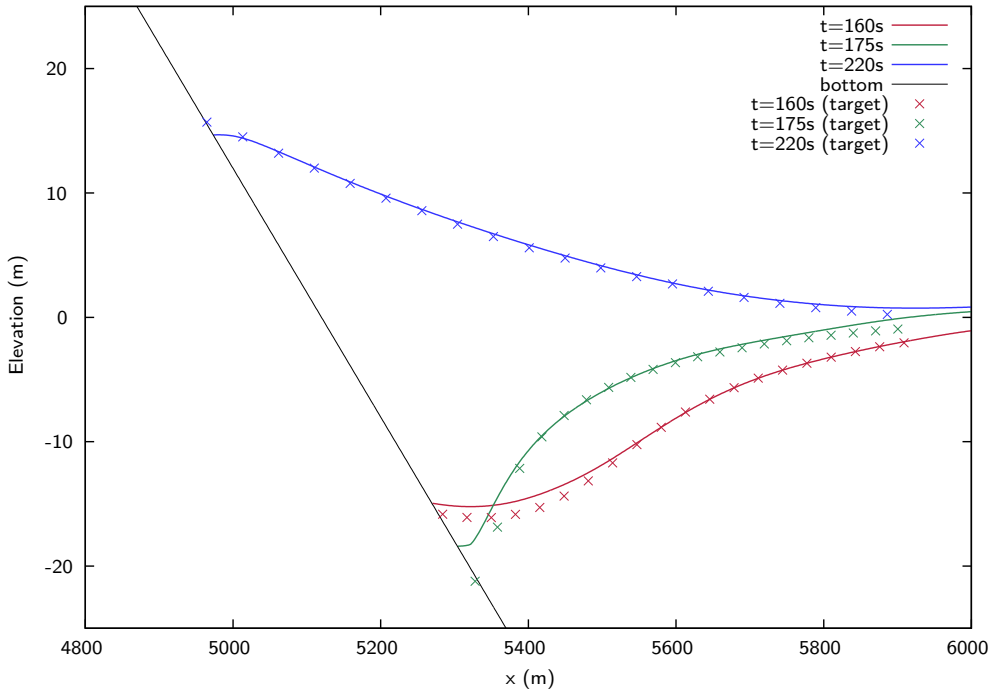


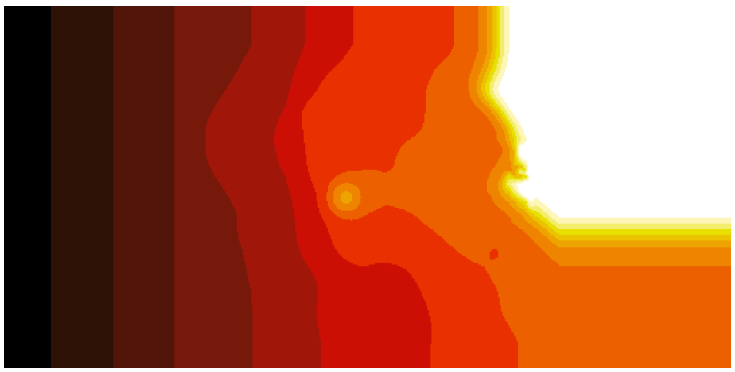
Figure 5.6: Free surface in the runup region

Tsunami runup onto a complex three-dimensional beach

Finally, a real-world application for the shallow water equations is examined and simulation results are compared to experimental data. In 1993, the Okushiri tsunami caused unexpected and desastrous damage, and at the same time provided numerous field data to check and benchmark tsunami runup models. A comparable benchmark problem was set up and a $1/400$ scale experiment of the Monai runup was conducted. These experiments were part of the set of benchmark problems used in the 2004 Catalina Island, Los Angeles, California NSF long-wave runup models workshop (Liu et al., 2008) and are used again in 2010s PMEL workshop.

Parameter	Value
Domain	5.488m $\times$ 3.416m
Grid	392 $\times$ 244
Δx	0.014m
c	56 m/s
Δt	0.25ms
$1/\tau$	1.8
δ	0.00001
BC x	Wave height
	No-slip
BC y	No-slip

(a) Parameters



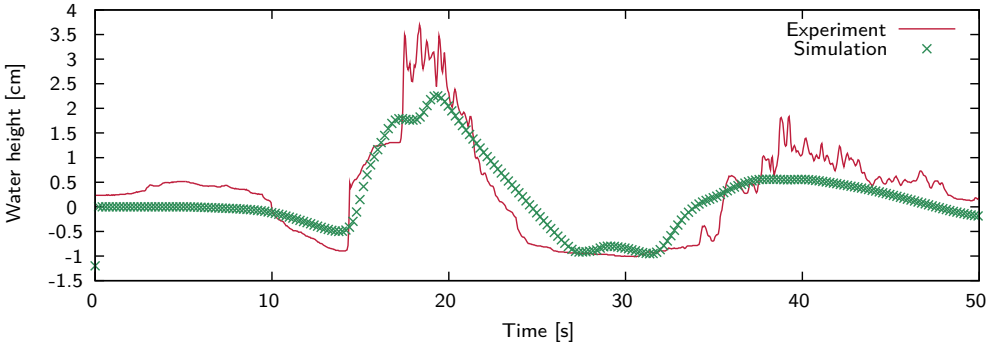
(b) Coastal topography

Test case 5.5: Tsunami run-up onto a complex three-dimensional beach

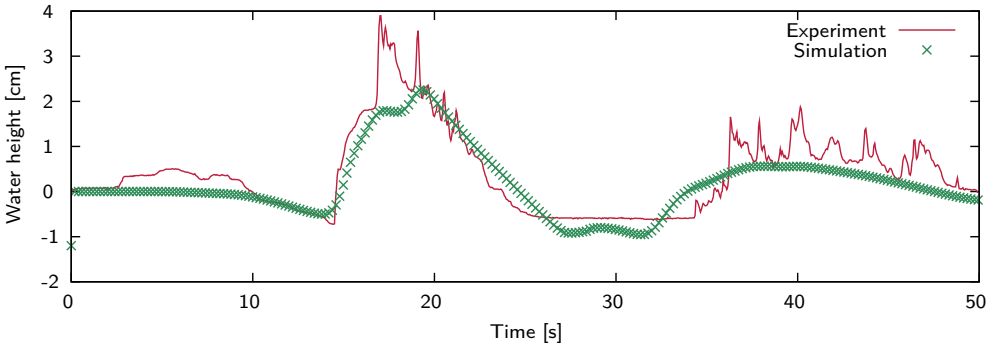
The scale model dimensions and parameters are given in test case setup 5.5. The domain of 5.4m times 3.4m is discretized according to the benchmark setup, proposing a grid spacing $\Delta x = 1.4cm$. The time step is chosen as $\Delta t = 0.25ms$, which directly sets the macroscopic reference velocity to a value of $c = 56m/s$. The bathymetry and the coastal topography are given below. The time-dependant offshore wave height (specified at the $x = 0$ boundary) is given. The test case is especially demanding in terms of bottom elevation. The forcing terms have to be evaluated for all lattice directions. Moreover, the runup model has to be extended to cope with non-axis-aligned changes of bottom topology.

In Fig. 5.7, the results for three different wave gages are compared to experimental data from the scale experiment. The general flow behavior is well represented, and the numerical predictions for the time of the initial impact and the arrival time of the reflected wave match the experimental values quite well. The maximum water heights of the numerical probes, however, are approximately 20% below the experimental values. Attempts to further reduce the fluid viscosity to capture the water height peaks induced instabilities in the runup model. Consequently, the development of a more complex wetting and drying algorithm for the simulation of runup on a complex bottom topography will be part of further research.

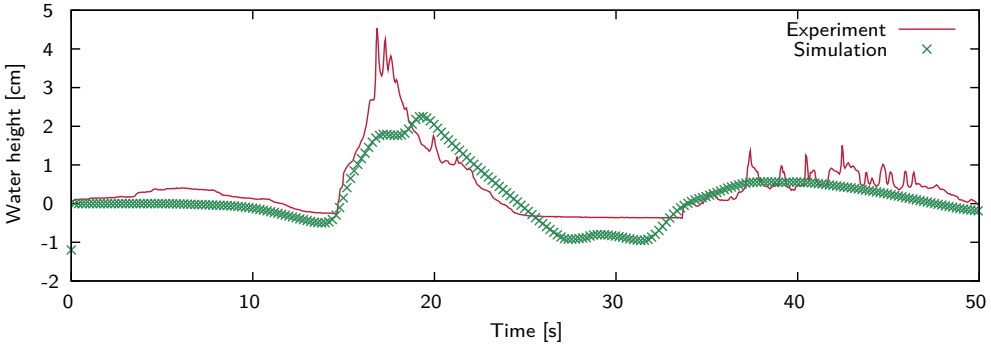
The simulation was carried out at a maximum performance of 250 MNUPS on a Tesla c1060 GPU. For the time period of 50s (corresponding to 200,000 time steps) on a lattice with 512x256 nodes, 100 seconds of computational time are needed.



(a) Probe A, $\mathbf{x}_A = (4.521m, 1.196m)$



(b) Probe B, $\mathbf{x}_B = (4.521m, 1.696m)$



(c) Probe C, $\mathbf{x}_C = (4.521m, 2.196m)$

Figure 5.7: Comparison of experimental and numerical results for three different wave gaves A,B and C for the Catalina 2 benchmark

5.4 Free surface kernel

Finally, the performance and validity of the free surface extension is investigated. It is based on the LB bulk flow scheme and makes use of the previously shown D3Q19 kernel for the calculation of the flow field. The phase interface is captured with the VOF approach described in section 4.2. The resulting overall algorithm (Alg. 4.1) consists of two nearly independent parts: the flow field is calculated, then the advection equation is solved.

The free surface part of the algorithm is hard to optimize for GPU hardware, precisely because it is mainly non-local, the computational domain varies in time and the advection steps apply to the interface nodes only. The latter is considered via a supplementary flag field, which marks the interface nodes and is queried in the following interface-update kernels. Opposite to the collision-propagation routines, no thread synchronization is needed, because all kernels update local variables only and do not introduce non-local memory write accesses. The calculation of new fill levels is straightforward, although the evaluation of Eq. 4.23 requires neighbor information about the node state and, respectively, the fill level.

If the computational domain evolves in time, nodes change their state, from interface to fluid or gas, respectively. In order to ensure a closed interface layer, all surrounding nodes have to be checked and - if necessary - changed from gas to interface state. This non-local *write* operation has to be modified, as one can not guarantee exclusive memory access for one thread or thread block. Hence, the change of nodal states is transferred from a non-local write to a non-local read operation: all gas nodes check if they are in the vicinity of an interface node, and, if yes, if this interface node changed its state to fluid. In analogy, fluid nodes near emptying interface nodes execute a similar algorithm.

The initialization routine of new fluid nodes is split up into two parts. First, a valid set of particle distribution functions is prescribed for all new fluid nodes, according to Eq. 4.25. For the interpolation of macroscopic values, only existing neighboring fluid nodes may be used. Thus, a second kernel switches the state of the new fluid nodes from gas to fluid once the initialization procedure is finished. As the interface evolves and new node states are set, lost interface cells might occur in underresolved areas. These interface cells without any fluid neighbors have to be detected and condensed.

The additional memory requirements are manageable: the interface is captured via a boolean flag matrix, which marks interface cells, two fill level matrices and a geo matrix to store the node state. The total memory costs for one LB node hence yield `(2*19+2*1)*sizeof(float)+1*sizeof(int)+1*sizeof(bool)` = 165 Byte.

5.4.1 Validation

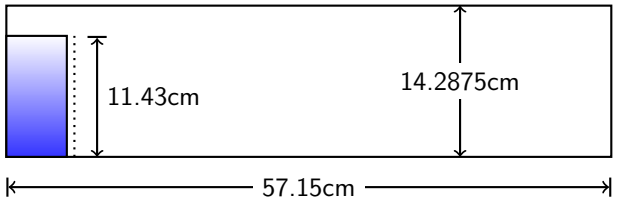
In the following, the numerical wave tank is analyzed and benchmarked with the help of well-known free surface test problems. The resulting performance drop in comparison to the LB bulk scheme is evaluated and analyzed.

Breaking dam benchmark

The classic breaking dam benchmark by Martin and Moyce [112] is used to demonstrate that the model is able to cope with real-world fluid simulations. A water column in a channel is constrained by a waxed paper diaphragm, see test case setup 5.6. Once the waxed paper is freed, the water column collapses. No-slip boundary conditions are used for all six walls. Martin and Moyce [112] determined a maximum dimensionless velocity of $u = 1.71$, which corresponds to Re ≈ 103483 and Fr ≈ 2.418. Viscosity ν and forcing g in the LB context are adjusted to match the given dimensionless numbers. The calculations are stopped when the surge front reaches the back wall of the container.

Param.	Value
Re	103483
Fr	2.418
u_{max}	1.71 [-]
Domain	0.5715m x $(0.142875\text{m})^2$
	$(22.5\text{in} \times 5.625\text{in}^2)$
Water column	2.25in x 4.5in x 5.625in
Lattice	128x32x32
	256x64x64
	384x96x96
	512x128x128

(a) Parameters

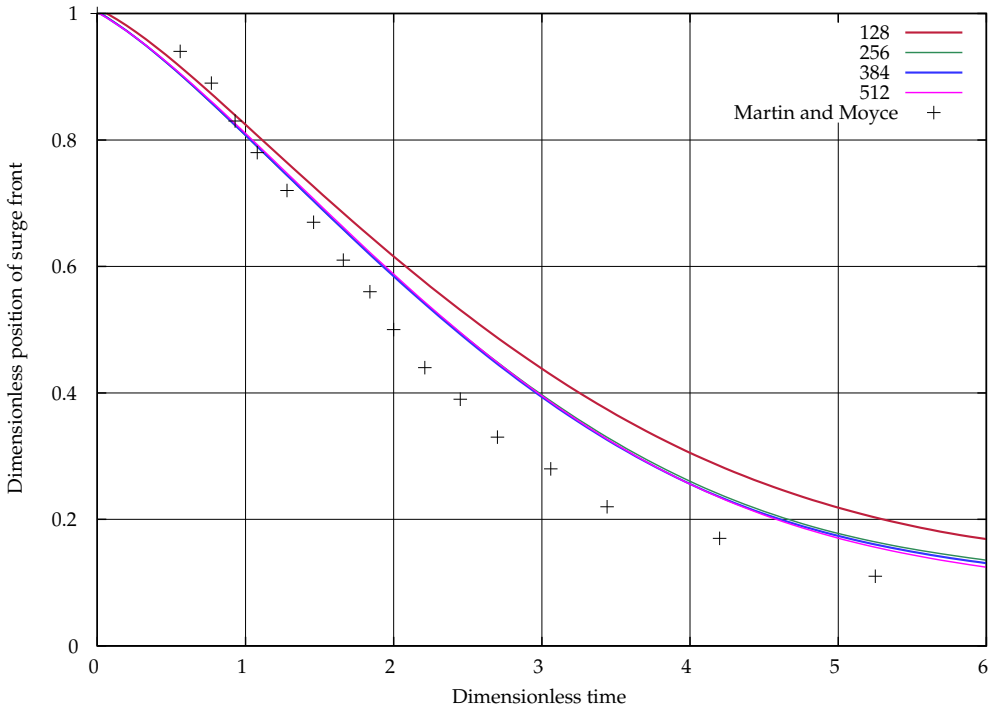


(b) Geometry

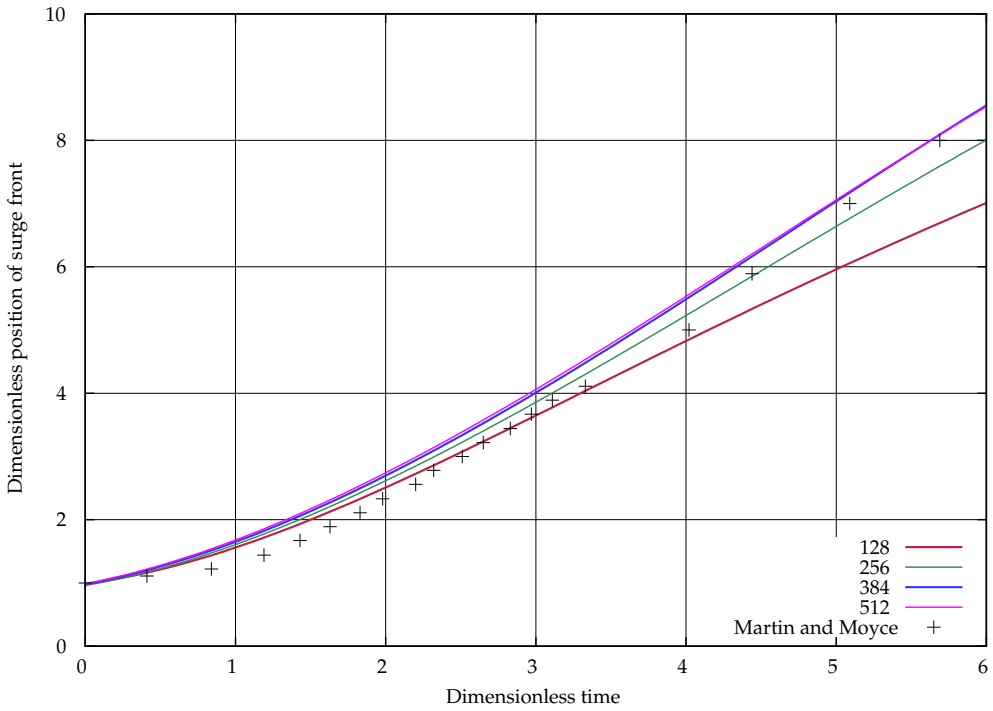
Test case 5.6: Breaking dam

During the simulation, the position of the surge front and the height of the collapsing water column are observed. In Fig. 5.8a and Fig. 5.8b the numerical results for four different grid resolutions are compared to the experimental reference data from Martin and Moyce [112]. Very good agreement and a convergent behaviour can be observed for both the surge front and the collapsing column. Our numerical surge front (Fig. 5.8b) evolves slightly faster than the one in the experiment. This might be due to the fact that the delay owing to the triggering (a thin diaphragm which is released by an electric current) is not modeled in our numerical wave tank. This effect also was observed by various other groups, e.g. Sauer [141], Kölke [99], Salih and Moulic [139]. Concerning the height of the collapsing water column, the initial high discrepancy decreases for higher resolutions (Fig. 5.8a). The remaining difference might be due to boundary conditions, as the water height is evaluated near the back wall of the channel. In section 7.4.1, the same breaking dam test case is examined with a different free surface advection scheme.

Apart from the numerical quality of the free surface model, the performance is of great interest. At first glance, it can be seen that the overall node udpate rate varies with time. In Fig. 5.9a, the performance is plotted over the dimensionless time. At the beginning of the simulation, a maximum performance of approximately 160 MNUPS for the finest grid can be measured, which increases up to 200 MNUPS as the water column collapses. The similarities of Fig. 5.9a and Fig. 7.5b are not a coincidence, when recalling the way the GPU deals with threads and blocks and warps. Basically, if a kernel which is executed on the GPU contains branching (as in if-statements), the whole kernel is executed multiple times, once for each possible branch. Moreover, threads are executed in warps



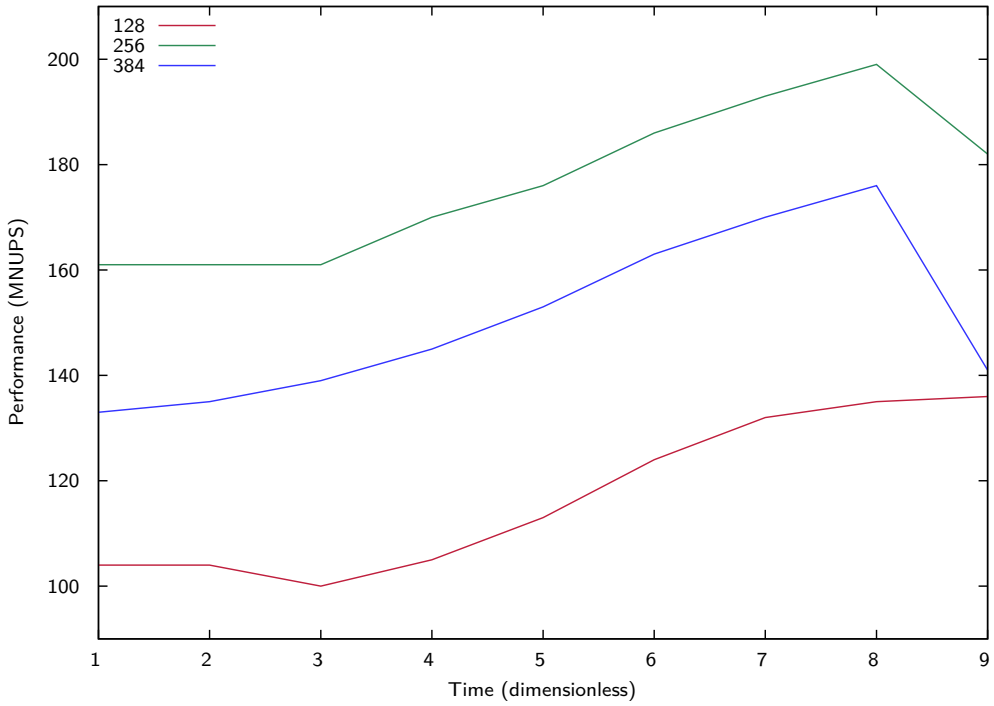
(a) Position of the water column top



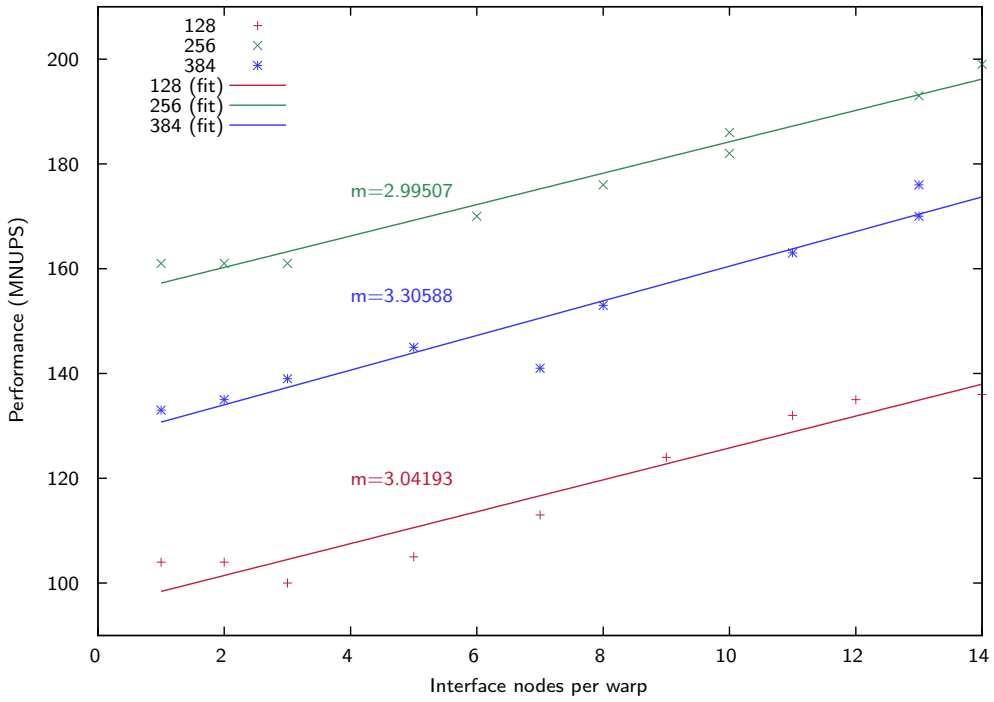
(b) Position of the surge front

Figure 5.8: Breaking dam benchmark, comparison of numerical and experimental results (GPU)

of 32 threads each, which are aligned along the x-axis in our grid mapping. Hence, if one warp of 32 threads contains a nonzero number of interface nodes, the whole warp executes the kernel twice: once for the normal fluid nodes (and the classical collision and propagation routines) and once for the interface nodes and the additional update interface algorithms. As a consequence, the warps only contain a small number of interface nodes due to the steep dam/wave front during the initial stages of the breaking dam simulation. As the water column collapses, the fluid is spread out, trying to establish a flat water surface, which would be in perfect alignment with the numerical grid mapping. The warps would be fully occupied with interface nodes. In practice, the optimal number of interface nodes (32) will not be reached due to local fluctuations of the water height. Fig. 5.9b depicts the relation of the number of interface nodes per warp to the overall performance of the algorithm, and a nearly linear dependency can be seen. Correspondingly, as the surge front impacts the right wall of the container at time step $t = 10$ in Fig. 5.9a, the performance drops immediately, as the average number of interface nodes per warp drops.



(a) Performance over time



(b) Performance over number of interface nodes per warp

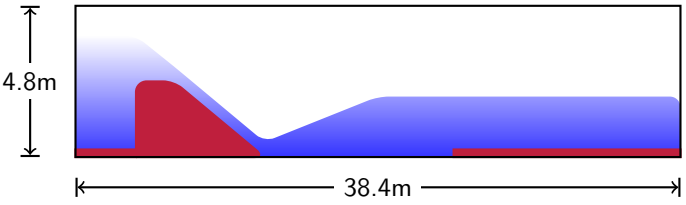
Figure 5.9: Breaking dam benchmark, performance of the GPU simulation

Hydraulic jump

As a first application for the GPU wave tank, the flow past a weir is examined (test case setup 5.7). We apply a velocity boundary condition on the left and a zero-gradient boundary condition on the right wall. No slip BCs are used at the bottom, slip BCs at the front and back wall of the domain. The parameters for our simulation are chosen in such a way that the subcritical inflow switches to a supercritical state while or shortly after passing the weir and leaves the domain again in a subcritical state.

Parameter	Value
Domain	38.40m x 6.40m x 4.80m
q	2.8 m^2/s
v_1	0.70 m/s
g	9.81 m/s^2
ν	1 · 10E-6 m^2/s
Lattice	192x32x24 ($\Delta x = 0.2m$)
	384x64x48 ($\Delta x = 0.1m$)
	576x96x72 ($\Delta x = 0.067m$)
	768x128x96 ($\Delta x = 0.05m$)

(a) Parameters



(b) Geometry (2D cut)

Test case 5.7: Flow past a weir

Such a properly designed hydraulic jump is used in civil engineering as a mean to dissipate energy and reduce erosion effects in the river bed. The corresponding water heights to generate such a scenario can be calculated via Bernoulli's equation, yielding

$$h_2 = \frac{1}{4} h_1 \mathrm{Fr}_1^2 \left(\sqrt{1 + 8\mathrm{Fr}_1^{-2}} \right) \tag{5.7}$$

$$h_3 = \frac{1}{2} h_2 \left(\sqrt{8\mathrm{Fr}_1^2 + 1} - 1 \right) \tag{5.8}$$

where the indices denote the inflow (1), the area directly behind the weir (2) and the outflow region (3). The corresponding values of Froude number and velocity in the sections 1 - 3 of the weir are given in Fig. 5.11b. The Froude number Fr_2 behind the weir also determines the stability of the hydraulic jump, see e.g. te Chow [154]. For Froude numbers around 4.5 we expect an oscillating and wavy hydraulic jump. In order to guarantee a minimum flux past the weir, the weir dimenions have to be set correctly. Following Poleni, the average flux q (per unit width) past a weir can be calculated via

$$q = \frac{2}{3} \mu \sqrt{2g} h_w^{1.5} \tag{5.9}$$

with gravity g, water height h_w above the top of the weir and a shape parameter μ which accounts for the weir shape. The weir geometry has to be chosen carefully, to guarantee the required performance and to avoid negative pressures and the risk of cavitation. A good hydraulic weir profile with a shape parameter $\mu_{WES} \approx 0.7$ is the *WES profile*, which is described by the following relation:

$$z_{weir} = \begin{cases} x^{1.85} / (2h_d^{0.85}) & : x < x_t \\ \tan(\alpha)\, x & : x > x_t \end{cases}, \tag{5.10}$$

with $x_t = h_w \cdot 1.0961 \cdot \tan(\alpha)^{1.1765}$ and the weir inclination α. With the given incoming flux $q = 2.8m^2/s$, Eq. 5.9 yields a water level above the weir top of $h_w \approx 1.20m$ (Fig. 5.11a) and hence a maximum weir height $h_{weir} = h_1 - h_w = 2.80m$. A step is installed in the rear part of the domain and prevents the hydraulic jump from moving downstream and leaving the domain. Similar steps are used in practice to stabilize hydraulic jumps.

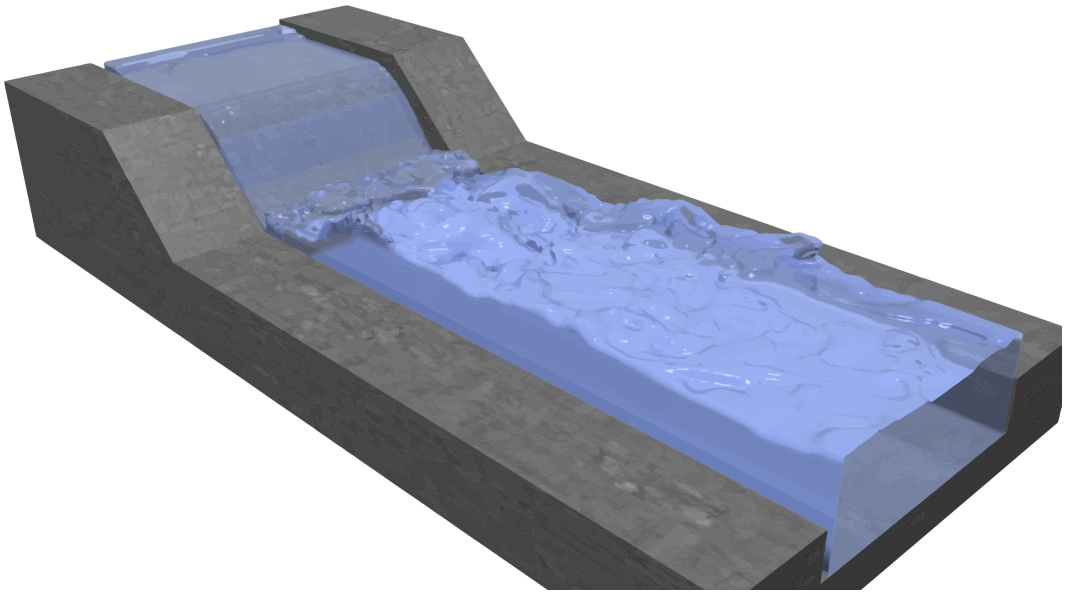
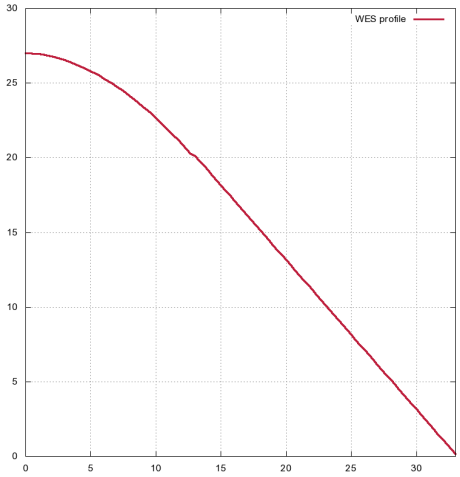


Figure 5.10: Hydraulic jump, $t = 60s$

A snapshot of the simulation after $t = 60s$ is shown in Fig. 5.10. The gauge heights and flow velocities in section 2 and 3 match the estimated values, see Fig. 5.11b. The gauge height on the back of the weir is higher than predicted. Note that the reference data is obtained from Bernoulli's equation, which is only valid along streamlines in inviscid flows and does, in addition, not consider three-dimensional turbulent effects. Hence, the reference data only can serve as an approximate value. Apart from that, the over-prediction of the water height h_2 might be due to the no-slip boundary condition on the weir surface and to the low resolution in this high-speed area of the flow. Grid refinement slightly improves the value of water height probe 2. In Fig. 5.11d, the x-component of the flow velocity in the hydraulic jump is shown. The maximum flow velocity does match the predicted value of $v_2 = 8.51m/s$, and the expected flow behavior in the hydraulic jump can be observed, including turbulent structures and local backflow. Moreover, the importance of the wall effects and the low-velocity region at the transition between the weir and the river bed can be seen. Finally, the Smagorinsky LES model tends to overestimate the turbulent effectsin the near-wall region, which should be tackled by the use of dynamic Smagorinsky models or an LES WALE approach, as discussed in section 3.7.

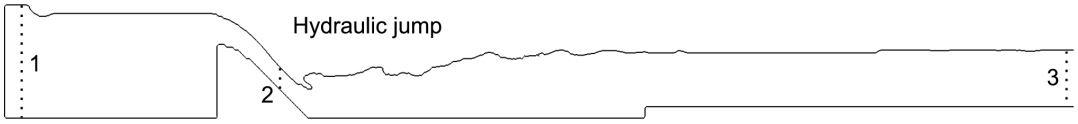
For this simulation, we obtain an average performance of 55 MNUPS on a GTX 275 for the coarse grids ($\Delta x = 0.1m$ and a time step of $\Delta t = 1.4 \cdot 10^{-3}s$). For the high-resolution simulation with a grid spacing of $\Delta x = 0.05m$ and a total number of 9.5 million nodes, a modern C1060 card is used. The average performance for the 60s-simulation yields 130 MNUPS.



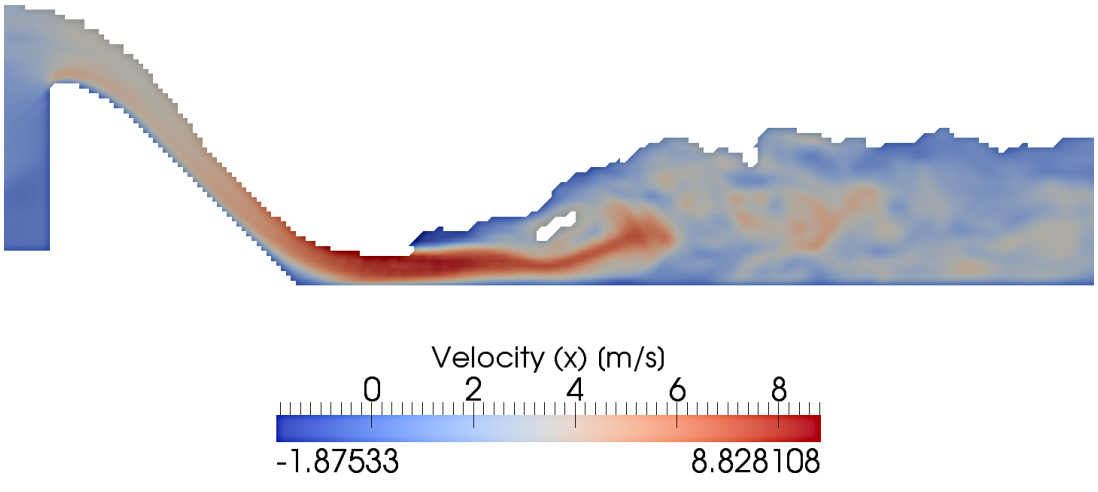
(a) Shape of WES profile

Section	1	2	3
Fr	0.11	4.74	0.31
v	0.70	8.51	1.37
h_{theory}	4.0	0.33	2.05
h_{192}	4.0	0.9	2.00
h_{384}	4.0	0.78	2.00
h_{576}	4.0	0.78	1.93
h_{768}	4.0	0.69	1.90

(b) Parameters in sec. 1 to 3



(c) Hydraulic jump (slice) with location of the sections 1-3



(d) Velocity profile (slice, nx=576)

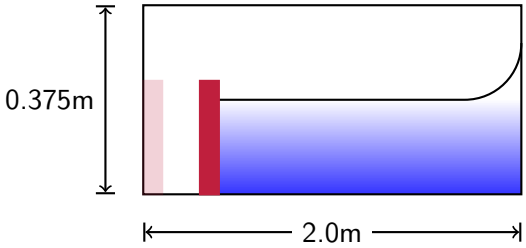
Figure 5.11: Results for the flow past a weir (GPU simulation)

Wave runup

A typical test scenario in experimental wave tanks is the study of wave runup on vertical walls, as e.g. shown in test case setup 5.8. Solitary waves are generated by flap or piston-type wavemakers at the left wall of the flume. The wave propagates towards the opposite wall and runs up the vertical wall. Similar experiments e.g. have been carried out by Gotoh et al. [60], who also compare the experimental results to numerical solutions, whereas Fenton and Rienecker [37] use a Fourier method to obtain results for the runup height.

Parameter	Value
Domain	2.0m x 0.375m x 0.375m
g	9.81 m/s^2
ν	1 · 10E-6 m^2/s
D_{water}	2.0m
Lattice	512x96x96

(a) Parameters



(b) Geometry

Test case 5.8: Wave Runup at Upright Wall Without Overtopping

In the following, we establish an enhanced numerical GPU wave tank and try to reproduce the experimental results of Gotoh et al. [60]. For most of the engineering applications in a numerical wave tank, wave generators are needed to generate meaningful free surface profiles to reproduce wave impact events. From the LB point of view, a wavemaker represents a moving boundary. In each time step, the wavemaker geometry is updated and lattice nodes are activated and deactivated, and a no-slip boundary condition with additional terms for the boundary velocity is used.

For benchmarking the wave generation, a solitary wave of height $0.24m$ is generated by means of a Goring-type piston wave maker and propagates in water of depth $d = 0.8m$. The overall flume is $40m$ long. The water height in the numerical wave tank is observed at eight probe positions at $x = 4m$ - $x = 32m$. In Fig. 5.12 the numerical wave height below the solitary wave is compared to experimental results of Yim et al. [177] which have been obtained in a 40m real-world wave tank at probe location $x = 15.7m$. The initial results show very good agreement with the experimental data, both for the peak elevation and the decay behaviour. However, during the wave propagation, the wave amplitude decreases due to dissipative effects, which is a known issue of methods on the basis of viscous flows. The initial large amplitude drop can be traced back to the simplified VOF surface reconstruction scheme which is used in the GPU wave wavetank, and which is the motivation for the development of an enhanced, hybrid VOF method in Part III of this work. The wave shape changes, but then is advected properly. Nonetheless, for this runup test case with an overall small domain length of $2m$ the wave generation is sufficiently accurate.

The runup test case is run for waves in water of depth $D = 0.2m$ and relative wave heights H/D from 0.1 to 0.4. In Fig. 5.13 our results are compared to the results of Fenton and Rienecker [37] and the experimental results of Gotoh et al. [60]. Very good agreement can be observed for small wave amplitudes, whereas for higher ones, the present model underestimates the vertical runup. Main drawback apart from the noticeable dissipation is the Goring-solution for driving the piston wavemaker at the offshore wall of the wave tank. It has shown to be only sufficiently accurate for

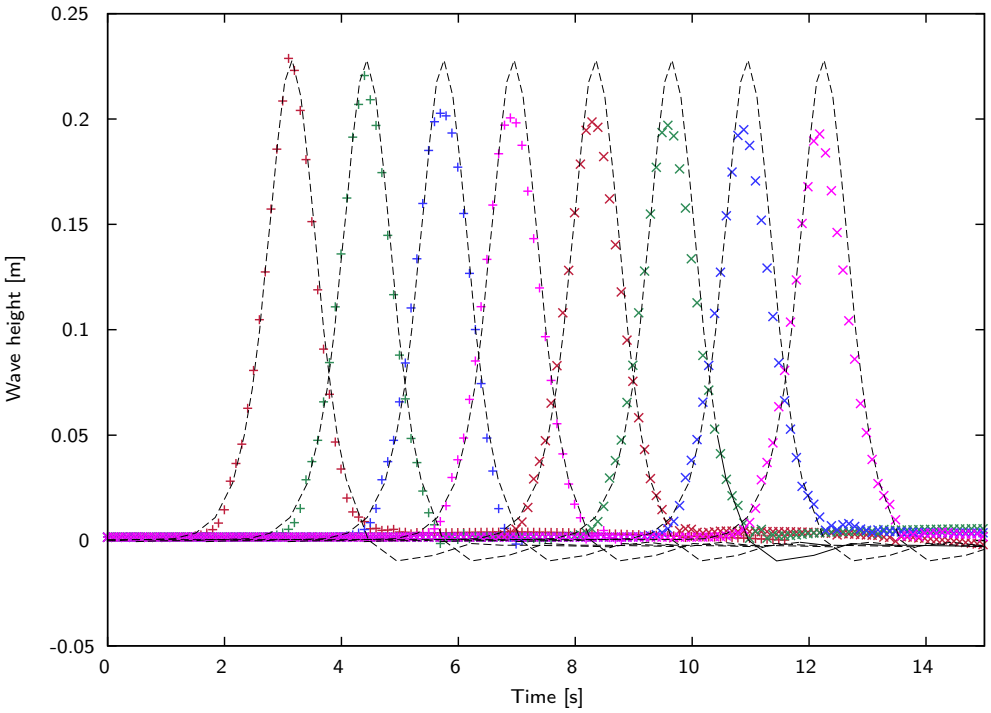


Figure 5.12: Solitary wave propagation: experiment vs. numerical solution, observed at eight probe locations ($x = 4, 8, .., 32m$, GPU simulation)

relative wave heights below or equal 0.2. Hence, for higher waves, as present in the current test case, more sophisticated wave generator theories have to be implemented in future work. Gotoh et al. [60] for example use the theory of non-reflecting wavemakers proposed by Hirakuchi et al. [82], in which the actual water heights in front of the wavemaker paddle iteratively influence the wavemaker motion. As an alternative for the wave generation, approaches for the initialization of the Lattice Boltzmann grid with very accurate results which are generated in numerical wave tanks on the basis of potential flow theory are discussed in chapter 8.

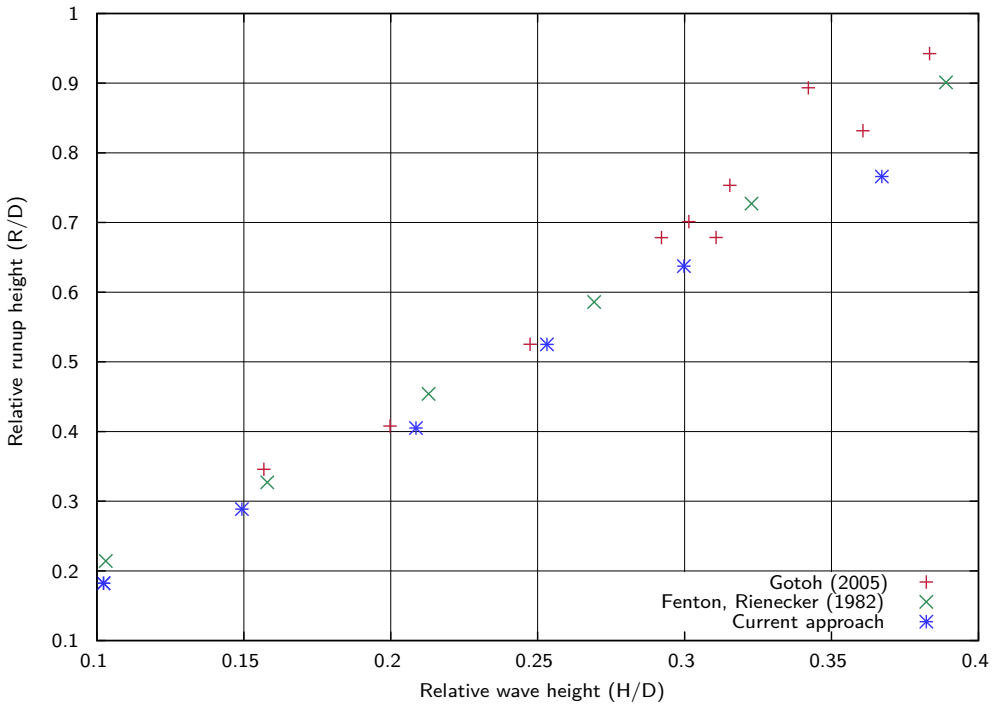


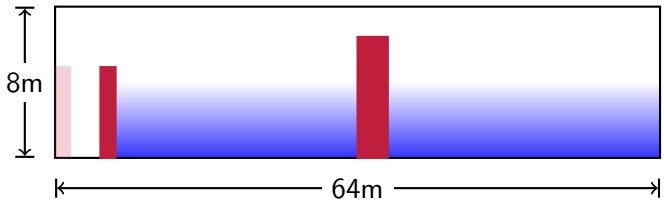
Figure 5.13: Results for vertical runup

Wave impact on cylinder

Finally, the algorithm is applied to a classical benchmark in the field of civil engineering, the impact of a breaking wave on a cylinder in the flow. Several experiments have been carried out to estimate the resulting slamming force on slender structures. Wienke [175] analyses the wave impact on cylinders for several different breaking and non-breaking waves and cylinder inclinations.

Parameter	Value
Domain	64m x 8m x 8m
g	9.81 m/s^2
ν	1 · 10E-6 m^2/s
D_{water}	4.25m
Lattice	512x64x64

(a) Parameters



(b) Geometry (2D cut)

Test case 5.9: Wave tank setup

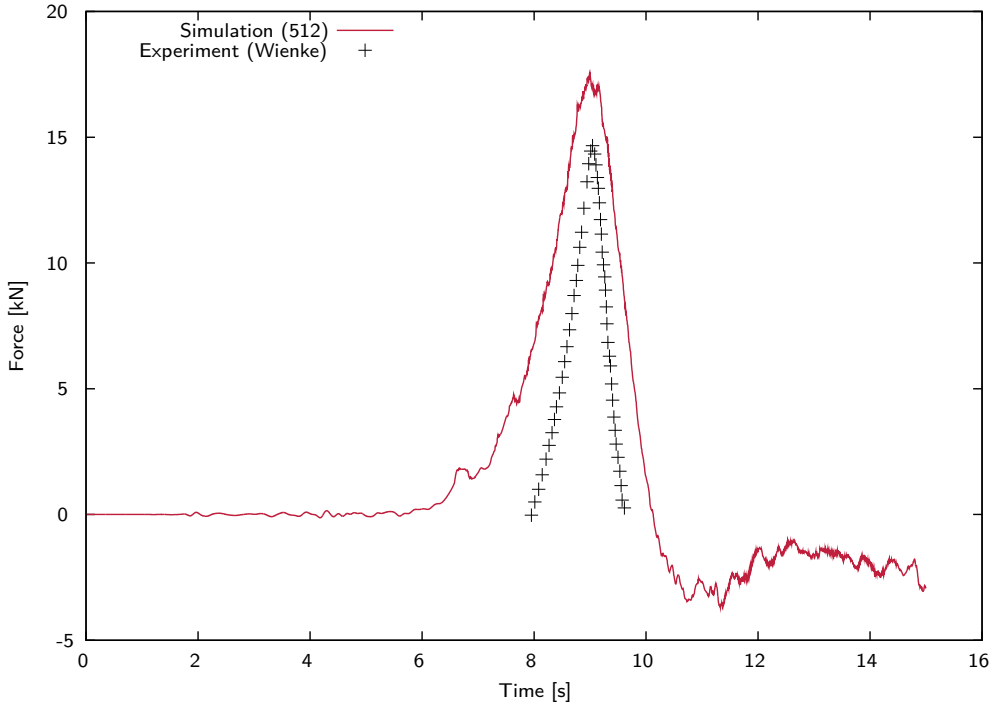
The geometry and simulation parameters are given in test case setup 5.9. At the left end of the domain, a piston wave maker serves to create the desired wave profile. In the study of Wienke, a mean water level of $D = 4.25m$ and a wave height $H = 1.5m$ are used. To account for the dissipative effects that were observed in the previous test case, the actual wave height in the vicinity of the cylinder is regarded and serves to adjust the wave maker target height. The 64m long wave channel has been discretized with 512 nodes, yielding a grid spacing $\Delta x = 0.125m$. The Mach number is set to $1/10$ and $1/100$, respectively.

For a non-breaking wave, the resulting total force on the cylinder is given in Fig. 5.14, in comparison to experimental data. The force on the cylinder rises continuously and no distinct force peak is present. As the wave passes by, the force decreases. The magnitude of the force matches the experimental data up to an error of 20%, and the time-dependant force evolution also corresponds to the experimental data. The discrepancies in the time signal are caused by a difference in wave types. Wienke [175] generates Gaussian wave packets, opposite to the solitary waves in the numerical wave tank. Nevertheless, both the wave heights during impact and the wave celerity $c = \sqrt{g(h+H)} \approx 7.5m/s$ match the experimental values, so that at least for the non-breaking case the comparison is valid. The final negative force on the cylinder is not reproduced by the experimental results, as, however, the time series provided by Wienke [175] does not extensively cover the time period after the wave impact.

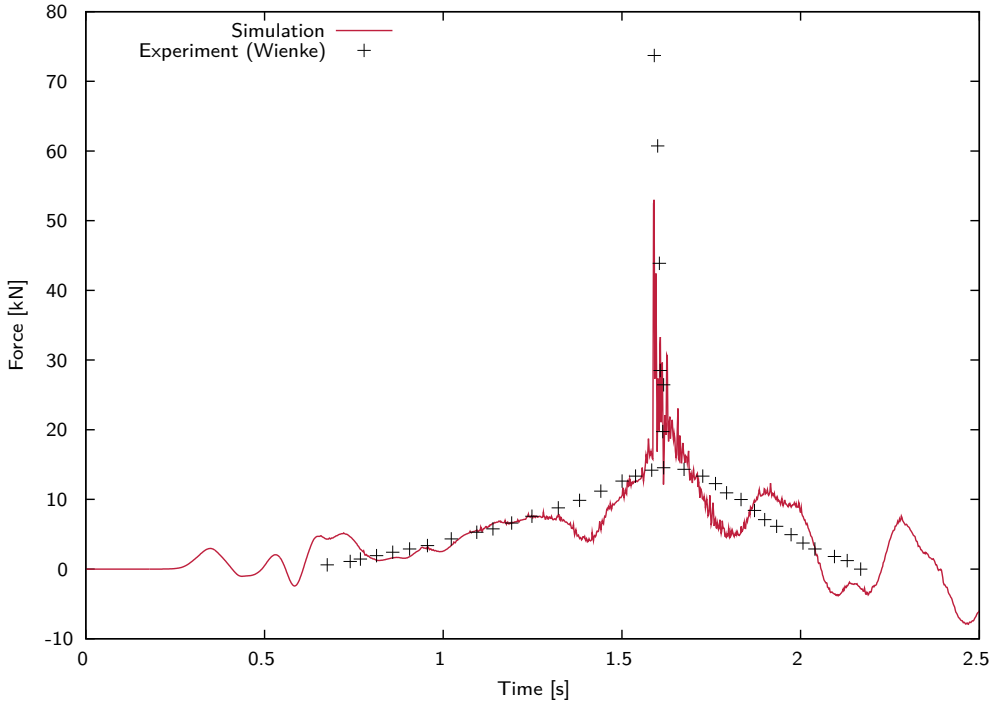
The force signals for two different Smagorinsky "constants" $C_s = 0.10$ and $C_s = 0.18$ are compared in Fig. 5.15. It can be observed that the force is overestimated for the higher value due to the overprediction of viscous effects in the near-wall regime. The boundary layer thickness is too high, which leads to an increased effective cylinder diameter and an increased force magnitude. Hence wall functions or wall-adaptive LES models should be used in future work.

It should be noted that the test case is exceptionally computationally expensive: the wave maker position has to be calculated and updated for every time step, leading to changes of the computational domain and of the boundary velocity at the left wall. On top, the drag force on the cylinder is evaluated each time step, to have a continuous solution control. Despite these additional computational operations, the average performance for this simulation yields 124 MNUPS on a GTX 275 graphics card.

For a maximum Mach number of Ma $= 0.1$, the resulting time step yields $\Delta t = 0.00360$s, corresponding to 4400 time steps for 15 seconds simulation. Hence, the whole simulation took approximately 1 minute. The time scales of real-world experiment and numerical simulations only differ by one order of magnitude. Note that in LB simulations, the time step Δt is related to the maximum Mach number of the flow. For simulations of the impact of breaking waves, the Mach number definitely will have to be lowered in order to obtain proper results for the force peaks, which will increase the computational costs. Fig. 5.14b compares the resulting force on the cylinder for a breaking wave to experimental data [175]. The results are preliminary, since the breaking wave is generated by a modified breaking-dam-scenario. For a detailed comparison and analysis of breaking-wave-impact, more sophisticated wave generation schemes are mandatory and the Gaussian wave packets of the experimental work of Wienke [175] have to be reproduced. Alternatively, a sophisticated initialization of the LB domain, e.g. by the results of a potential flow simulation, would provide adequate wave profiles, see chapter 8. Further investigations of the wave impact on structures is given in section 7.4.3.



(a) Non-breaking case



(b) Breaking case

Figure 5.14: Force on cylinder in comparison to experimental results of Wienke [175]

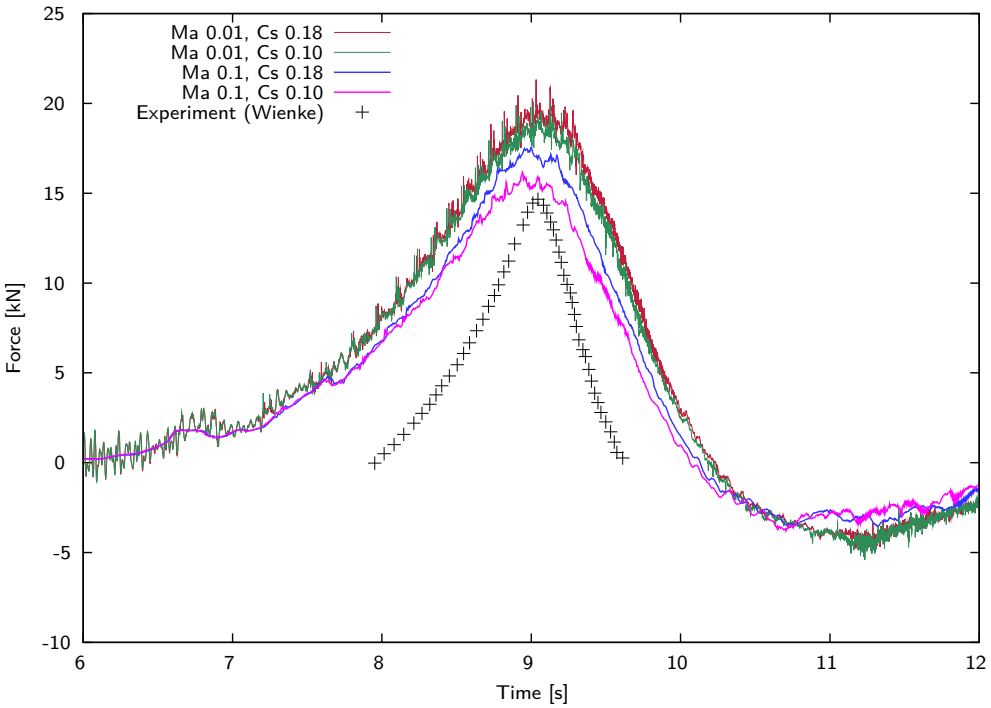


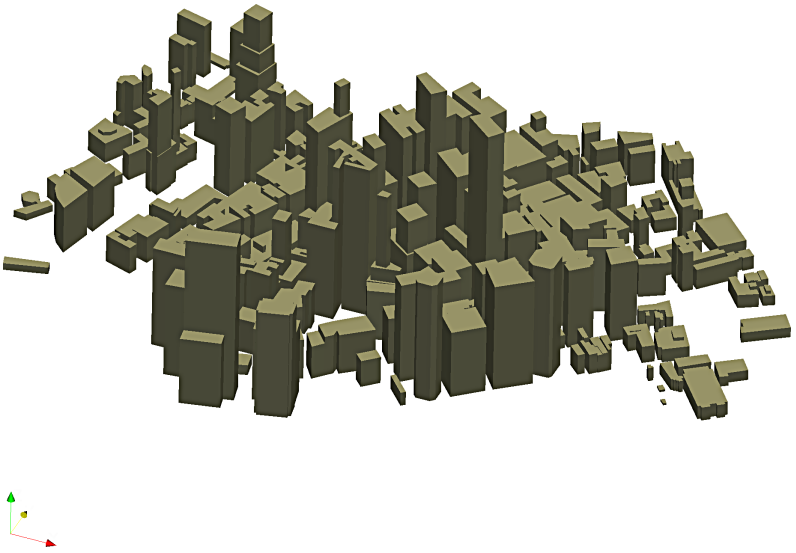
Figure 5.15: Non-breaking wave impact for two selected Mach numbers and two values for Smagorinksy constant C_s in comparison to reference data [175]

5.4.2 Wave impact on South Manhattan

Wave impact on engineering structures is one of the most popular applications of free surface flow solvers, as shown for the wave impact study on cylinders in the previous test case. Apart from single obstacles in the flow, scenarios which involve more complete and complex geometries are of great interest. From the numerical point of view, these test cases are demanding due to high Reynolds numbers and large compuational domains, which ask for massively parallel computing to tackle the simulations. The impact of a wave on South Manhattan (test case setup 5.10) is shown in order to demonstrate the capability of the GPU free surface solver to deal with complex geometries of large scale. A 1:10 scale model is used and discretized with a grid spacing of $\Delta x = 0.23m$. The wave is generated by a breaking dam scenario. Results for this simulation are given in Fig. 5.17 and Fig. 5.18. It can be observed that, even with a real world grid spacing of $\Delta x = 2.3m$, detailed information on the liquid entry after the initial wave impact can be obtained. The simulation was run on a nVIDIA Tesla C1060 card, using the maximum available device memory of 4 GB. In the medium term, the parallelization of the GPU wavetank to access the compute power of the 96 C1060 cards of LUDWIG would allow to quadruplicate the grid resolution for this simulation, leading to a grid spacing of less than 60cm.

Param.	Value
Domain	120m,120m,24m (1:10 scale model)
Lattice	256x256x48 512x512x80
Δx	0.47m, 0.23m (1:10 scale model)

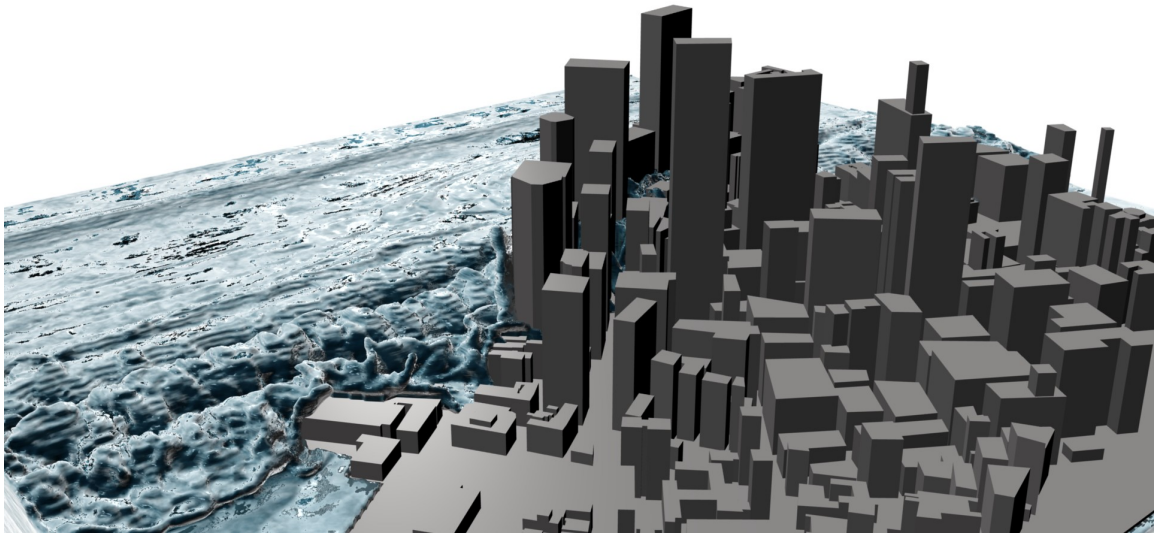
(a) Parameters



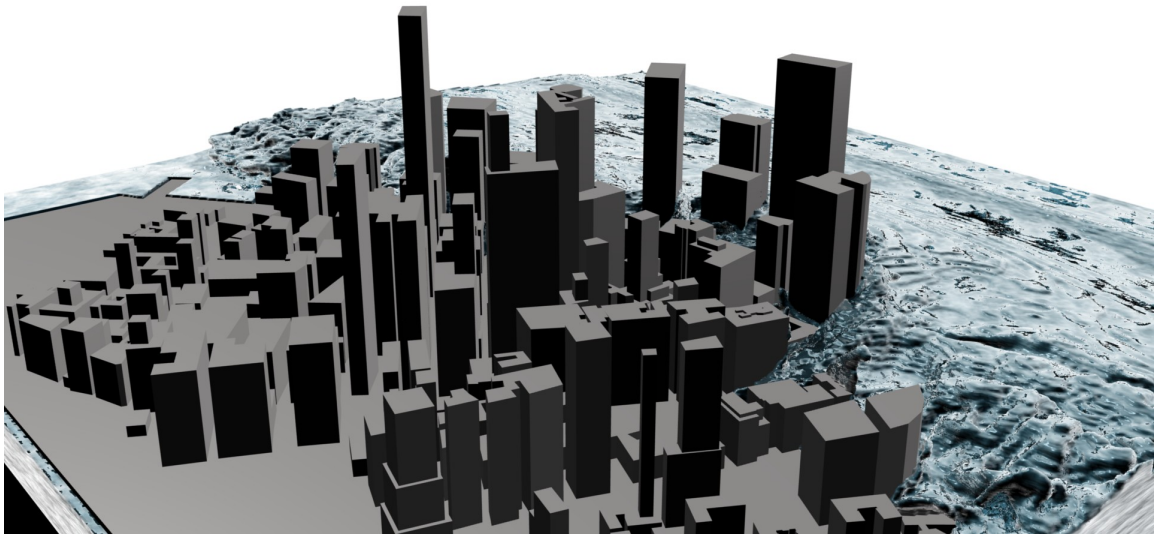
(b) Geometry

Test case 5.10: Wave impact on South Manhattan

Figure 5.16: View on South Manhattan, taken on Staten Island Ferry

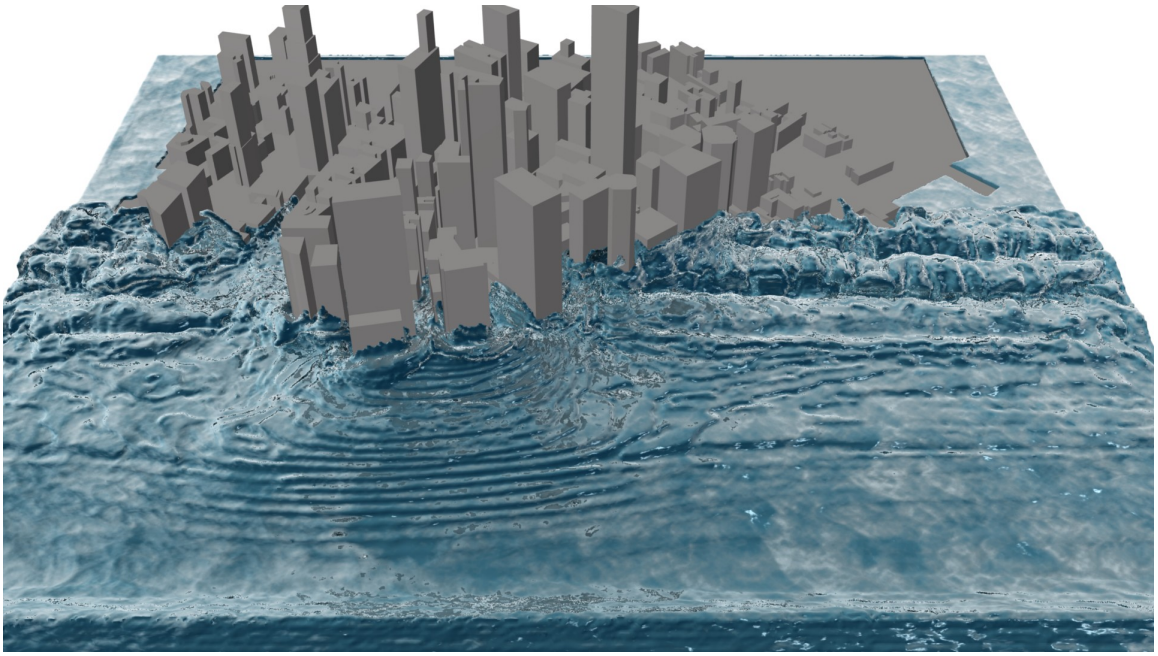


(a) North west view

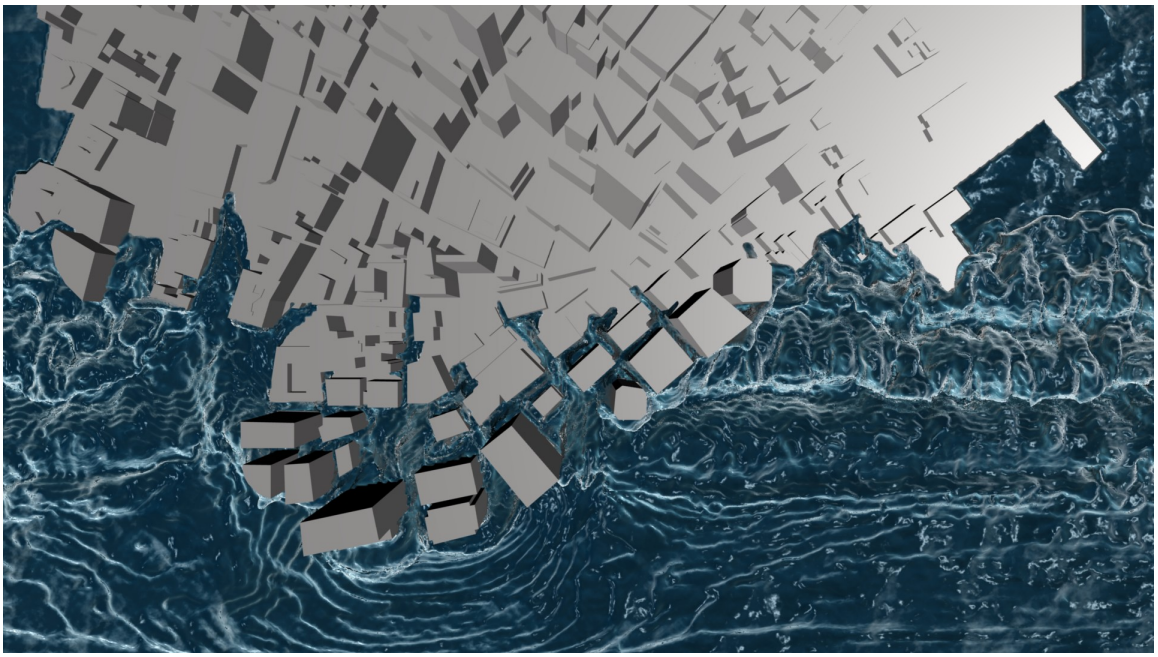


(b) North east view

Figure 5.17: Wave impact on South Manhattan, results



(a) South view



(b) Top view

Figure 5.18: Wave impact on South Manhattan, results (Ctd.)

5.5 Conclusions

In the previous chapter, we presented the GPU implementation of existing free surface and shallow water schemes on the basis of two- and three-dimensional LBM models.

The LB shallow water model has shown to be able to cope with real-world fluid simulations and up-to-date benchmark test cases of the tsunami community. Accurate results were obtained at very competitive runtime. For the wave runup, Boussinesq-type models are widely used nowadays, as they include the effects of dispersion which are crucial for runup studies. However, this drawback of a shallow water-based model was not observed in the Catalina runup benchmark and the results fit well. The runup study for the Monai valley experiment did expose several model drawbacks, especially in terms of the runup model. Nonetheless, the resulting performance is still very high, despite the topological changes during runup and the additional non-local memory access pattern of the shoreline algorithm. The quality of the results can be improved by using more sophisticated runup models, which would only have a minor impact on the overall performance. The general applicability of the LB shallow water model to various benchmark problems has been widely shown in the literature.

In 3D, several GPU-accelerated simulations of turbulent free surface benchmarks and applications were presented. The non-local operations in the free surface extension lead to a performance drop, compared to the stand-alone LB scheme. Nevertheless, the numerical wave tank ends up with approximately 20% of the performance of the bulk scheme in the worst case and up to 50% in the best case. The fluctuating performance drop is related to the alignment of the water surface. Most of the typical free surface flow simulations in a numerical wave tank start from a state of rest, with only small deviations from a flat water surface, so that this limitation seems to be acceptable. For free surface flow problems with very large topological changes, which do not involve flat surfaces, improved grid mapping strategies might lead to higher peak performance. However, the GPU implementation is still at least one order of magnitude faster than comparable CPU implementations. For the simulation of the flow past a weir with a total of 1.2 million lattice nodes and a corresponding time step of $1.4 \cdot 10^{-3}$s, a maximum performance of 55 MNUPS is achieved. Consequently, the calculation of 1 second of real world fluid behavior requires only 15 seconds on one single GPU in an ordinary workstation. The main drawbacks of the implementation are the strict limitations for the dimensions of the test case. Moreover, the maximum number of nodes is currently limited by the device memory (1 resp. 4 GB only). Hence, in the future, a multi-GPU implementation has to be addressed, and the algorithm will be embedded in a computational steering environment for more convenient pre- and postprocessing.

Apart from the high performance of the kernel, we have shown that the free surface implementation is in general suitable for the simulation of typical free surface flow problems in civil engineering. Both the flow past a weir and the wave impact on a cylinder were successfully simulated. Nonetheless, for more complex free surface flows involving fluid-structure interaction, complex geometries and more demanding applications, a more sophisticated free surface model is needed, which is presented in Part III of this thesis.

Part III

Enhanced free surface model

CHAPTER 6

Generic numerical treatment of phase interfaces

From the numerical point of view, a free surface or any other phase interface represents a moving boundary. Compared to obstacles moving with a predefined velocity, the motion of this boundary is not prescribed and the interface is allowed to move freely, as long as the kinematic and dynamic boundary conditions are fulfilled. In Euler- and Navier-Stokes solvers, the kinematic boundary condition has to be fulfilled explicitly, opposite to e.g. shallow water equations where they are implicitly fulfilled. The kinematic boundary condition leads to advection rules for the phase interface, which guarantee that the interface is moving with the local fluid velocity. At the same time, the interface has to be kept sharp, although large deformations and even topological changes may occur and drops may leave or enter a closed surface and spray occurs.

The previously described free surface algorithm is based on a very pragmatic view on phase interfaces. It leads to good results, as long as only macroscopic fluid behavior is of interest, and the detailed flow structure in the vicinity of the interface is not important for the overall bulk flow behavior. Opposite to that, real world free surface flow applications involve complex interactions between the two phases, and regions of high curvature and high velocity, which both is very demanding for the free surface scheme. The careful development of advection schemes (independent on the underlying flow solver) is crucial for successful simulations. Hence, the next upcoming interim goal is to develop an advection algorithm, which advects the phase interface on the basis of an existing velocity field. In a second step (chapter 7), the coupling of this algorithm to a LB flow solver will be realized.

6.1 State of the art

Two main classes of interface methods can be distinguished, see Fig. 6.1. Interface tracking methods follow the free surface position explicitly. In contrast to that, interface capturing methods capture the position of the free surface implicitly. Both approaches reveal advantages and disadvantages, depending on the field of application and the flow solver which is used to solve the governing flow

equations. Recently, hybrid methods have gained in importance, as they combine "best of both worlds".

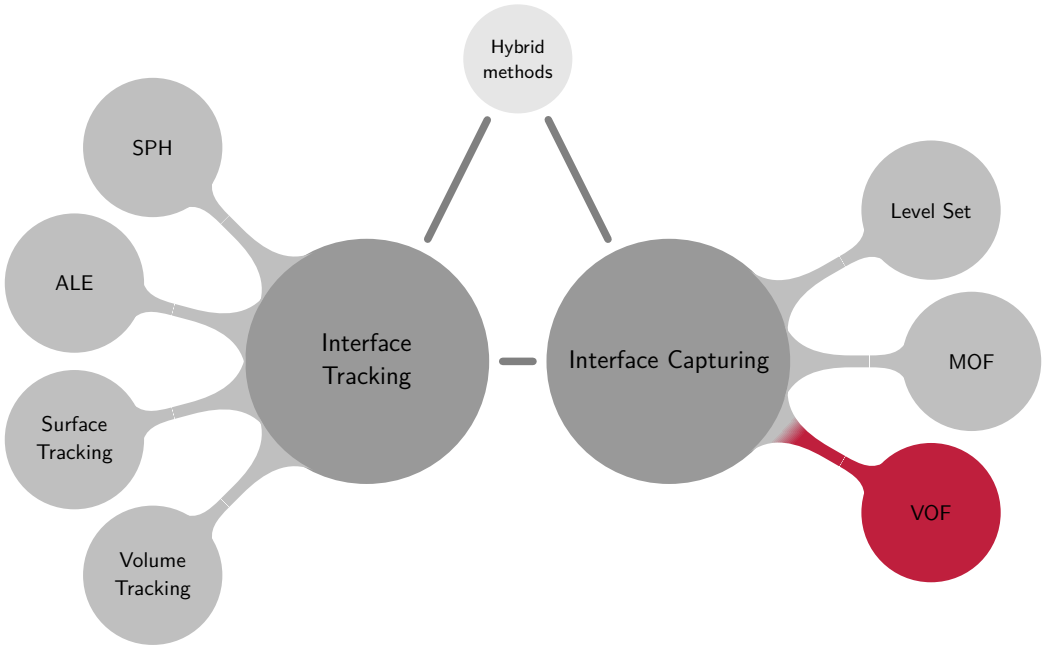


Figure 6.1: Interface capturing and tracking methods

6.1.1 Interface tracking methods

In interface tracking methods the location of the free surface is directly accessible and does not have to be recovered in a surface reconstruction step. In common *Lagrangian* interface tracking methods, the computational domain and the underlying discretization for the flow equations is changed in every time step to ensure a proper free surface representation (SPH, ALE). Opposite to that, in *Eulerian* tracking methods, the discretization of the flow field does not change in time, but the free surface is tracked by e.g. marker particles.

SPH

Smoothed-particle hydrodynamics (SPH) is a mesh-free method of Lagrangian type and is based on the work of Monaghan [118]. The fluid domain is discretized by a set of discrete particles, which have a spatial distance (known as the *smoothing length*), over which their properties are smeared by a kernel function. Hence, the physical quantities of any particle can be obtained by summing the corresponding properties of all the particles which lie within the range of the kernel. This way, numerous partial differential equations can be discretized and solved, e.g. the Navier-Stokes equations. The addition of free surface tracking is straightforward: a pressure boundary condition is applied at the free surface, whereas the kinematic free surface boundary condition is fulfilled by advecting (i.e. moving) the particles according to the particle velocity. Several SPH implementations have been developed and published. Open Source implementations for CPU clusters (SPHysics, [150])

and GPUs (gpuSPHysics, [23, 79]) exist. Due to the relatively expensive neighbor-search-algorithms, SPH methods spend most of the computational time on topological operations instead of solving the underlying equations. Main drawback of SPH methods is the treatment of pressure. To reduce the overall system stiffness, artificial compressibility often is introduced by lowering the speed of sound. Nevertheless, a noisy pressure signal, leading to stability issues, still is the main drawback of the method and asks for careful treatment.

Moreover, in SPH methods, the interface tracking is strongly coupled to the discretization of the flow equations and the SPH interface tracking can not be considered stand-alone. Mosqueira et al. [120] and Colagrossi and Landrini [19] apply the SPH to engineering applications such as rising bubbles and breaking dam benchmarks. Grenier et al. [61] extend the formulation and tackles the simulation of rising air bubbles, Rayleigh-Taylor instabilities and lock-release gravity currents cases, capturing effects of sloshing and mixing. More classical free surface benchmarks such as collapsing water columns are addressed by Roubtsova and Kahawita [137]. For the sake of completeness, we also refer to Dalrymple and Rogers [24], Colagrossi et al. [20], Kelager [92] and Marrone et al. [111].

ALE

The Arbitrary Lagrangian Eulerian (ALE) method is a second Lagrangian interface tracking method. In analogy to SPH methods, the computational domain follows the phase interface. The governing flow equations typically are solved in a Eulerian formulation. Then, the domain is updated in a Lagrangian way on the basis of the previously calculated velocity fields. Note that this holds for the domain boundaries (the free surface location) only. The internal mesh is adapted to the new free surface location in a second step. Often, linear-elasticity-based equations are solved for the mesh motion and the displacements of the phase interface serve as boundary conditions for the overall mesh deformation.

Methods on the basis of ALE formulations are applicable when no large topological changes are expected, as several crucial parameters concerning mesh skewness change during the mesh deformation step and topological changes would require remeshing or more intensive mesh optimizations. Moreover, the ALE allows for the monolithic coupling of fluid and structures and thus is well-suited for the simulation of composite free surface and fluid-structure interaction problems. Walhorn [169] applies the ALE method for the simulation of free surface flow problems and presents validations for dambreak and free jet scenarios, before the FSI extension is shown. However, no FSI problems involving free surfaces are demonstrated. In contrast, Tanaka and Kashiyama [153] proposes an extended formulation which also can handle hybrid FSI free surface problems.

Surface tracking

Surface tracking techniques identify the phase interface by a discrete set of points, which are advected in a Lagrangian way on the basis of a valid Eulerian flow field. The interface location is reconstructed as a set of continuous curves, which connects all the marker particles. Both the location and the correct order of the particles has to be stored. Glimm et al. [57] use a front-tracking approach for the simulation of Rayleigh -Taylor instabilities and extend the approach to three space dimensions (Glimm et al. [58]). Unverdi and Tryggvason [163] present both 2D and 3D algorithms for the simulation of

unsteady multiphase flow, including simulations of rising bubbles. An overview over surface tracking methods can be found in Shyy et al. [148].

Volume tracking (MAC)

As an extension to surface tracking, in volume tracking approaches, the particles are introduced to mark whole *areas* which contain fluid. This marker-and-cell method is going back to Harlow and Welch [73] and Welch et al. [172]. The marker particles are advected with the flow field in a Lagrangian way. Applied to free surface flow problems, the free surface is assumed to be located in those cells containing markers and having empty neighbor cells. For multiphase simulations, several kinds of particles may be introduced to distinguish between the phases. Theoretically, any number of different fluids can be treated. As the particles do not represent the interface directly, the method is considered to be a *volume* tracking scheme, as distinct from interface tracking in the proper sense. The MAC method has been applied to various single- and multiphase problems, e.g. the two-phase flow simulations of Rayleigh-Taylor instabilities (Daly [25]), the impact of solitary waves on vertical walls (Chan and Street [17]), or more recent work of Glowinski et al. [59] and Nakayama and Mori [122].

6.1.2 Interface capturing methods

Interface capturing methods introduce additional field variables to represent the phase interfaces, which have to be reconstructed out of this information. Additional variables are introduced for properties of the interface regions (fill level, center of mass, distance to the interface, ...). They obey advection equations and allow for a surface reconstruction step, in which the free surface position can be reconstructed up to a certain error.

VOF

In a volume of fluid (VOF) approach, a fluid fraction variable ε is introduced (see e.g. Hirt and Nichols [83]). It captures the *fill level* of a control volume V_{cv}, i.e. the volume fraction being filled with fluid:

$$\varepsilon(\Omega) = \frac{1}{|\Omega|}\int_{\Omega} dV = \frac{V_{fluid}}{V_{cv}} \tag{6.1}$$

A fill level of 0.0 marks a completely empty cell, a fill level of 1.0 marks a filled-up cell. A typical VOF algorithm consists of two steps: in the surface reconstruction step, the location of the interface is reconstructed out of the discrete fill level information. The resulting interface then is advected and the fill levels are updated. The main advantage of VOF methods is the inherent conservation of mass and the capability to deal with topological changes as drops leaving the domain or the merging of two sub-interfaces. On the other hand, the surface reconstruction is not straightforward, and interface diffusion is a critical issue to most VOF methods. A very detailed review of VOF methods will be given in section 6.1.3.

MOF

In a moment of fluid (MOF) approach, the first order moment of the free surface (i.e. the center of mass) is tracked in addition to the zeroth order moment (i.e. the fill level):

$$\mathbf{x}_\Omega = \frac{1}{\varepsilon(\Omega)} \int_\Omega \mathbf{x} dV \tag{6.2}$$

This allows for a local reconstruction of the surface normal. In a least-squares sense, the surface is repeatedly reconstructed until the target values for fill level and fluid centroid match. The moment-of-fluid method is using volume fractions as well as material centroids. Dyadechko and Shashkov [32] introduced the moment-of-fluid (MOF) interface reconstruction method, and successively extended it ([31] and [33]). Ahn and Shashkov [3] describe the extension to 3D for arbitrary polyhedral meshes and multi-material interfaces. MOF with adaptively refined grids on AMR type meshes recently has been introduced by Ahn and Shashkov [2].

Level Sets

In level set methods, the signed distance of a cell center to the free surface is introduced, see Osher and Sethian [128]. The free surface then is reconstructed as the zero-distance isosurface of the signed distance field. However, for the interface advection, no surface reconstruction is needed, but the motion of the surface is described by the advection equation

$$\frac{\partial \varphi}{\partial t} = v|\nabla \varphi| \tag{6.3}$$

which is similar to the underlying VOF equation, but this time describes the evolution of the signed distance function ϕ in time. For the evaluation of the phase field gradient, higher order schemes such as a 5th order WENO scheme is used. Main drawback of level set methods is that they do not necessarily guarantee conservation of mass. Recently, extensions have been proposed to cure this problem, often resulting in hybrid methods with a combination of Eulerian Level Sets and Lagrangian particle tracers [34, 7, 45, 46]. Also, hybrid LS-VOF methods (Ménard et al. [121]) can solve this problem. Here, a VOF method is used during the advection step, whereas the surface normal and surface curvature is obtained from a level set function. This method leads to mass-conserving and geometrically accurate results, at the price of high computational costs and memory consumption. Both underlying methods have to be implemented and additional correction terms serve to couple the methods.

6.1.3 Discretization: VOF models in detail

As mentioned above, a fluid fraction variable ε is introduced in VOF methods to capture the *fill level* of a cell. For the simulation of free surface flow in a VOF framework, an additional transport equation for the interface evolution between the two phases has to be solved. Hereby the main numerical difficulty is to maintain a constant width of the interface and to avoid artificial diffusion of the interface profile. The evolution of the fill level in time is described by the following advection

equation:

$$\frac{D\varepsilon}{Dt} = \frac{\partial \varepsilon}{\partial t} + \mathbf{v} \cdot \nabla(\varepsilon) = 0 \tag{6.4}$$

which is valid for a divergence-free velocity field $\mathbf{v}$. For the numerical solution, the advection equation for the VOF fill level ε has to be discretized. Various techniques have been applied, which are briefly reviewed in the following. However, all methods have their advantages and disadvantages. In any case, the solution of the advection equation is quite demanding, as the fill level is a discontinuous jump function.

Algebraic discretization of the advection equation

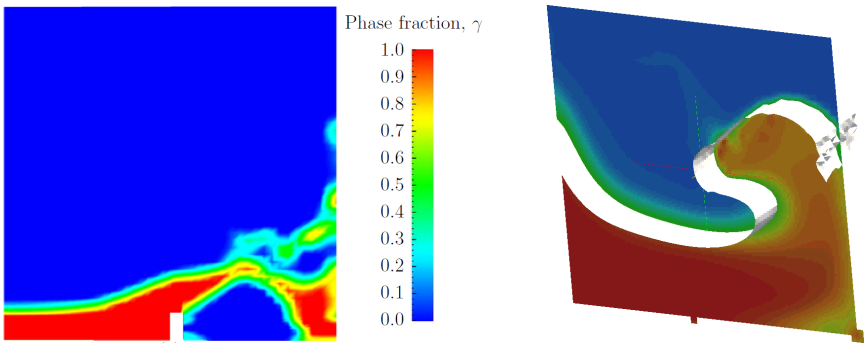
Classical finite-volume-based schemes discretize the convective term in the transport equation with a higher-order scheme and do not include a geometrical representation of the interface [104, 162]. The exact position of the free surface is not needed and no surface reconstruction takes place. The transport equation is evaluated for all cells in the computational domain, which may lead to cells with unphysical fill levels higher than one or lower than zero. The thickness of the phase interface is not limited explicitly, but diffusive interfaces appear, because - naturally - the gradient at the discontinuous fill level function is infinitely large.

The fluid fraction variable ε can be considered to be a jump function, jumping from zero to one at the phase interface. Hence, the advection schemes suffer from widely known problems in the CFD community for the treatment of e.g. pressure jumps, as they pose severe problems for compressible Navier-Stokes solvers in the transsonic regime, trying to capture acoustic shock waves. Hence, in past years, numerous mathematical methods have been developed to deal with this issue. Mathematically, the advection equation is a partial differential equation of hyperbolic type. In numerical methods, total variation diminishing (TVD) is a property of certain discretization schemes used to solve these hyperbolic partial differential equations. The concept of TVD was introduced by Harten [75]. According to *Godunov's theorem*, only first order linear schemes are TVD, although being numerically diffusive. All higher order linear schemes are not TVD and tend to introduce spurious oscillations where discontinuities or shocks arise. Often, limiter functions are introduced to switch between different approaches to avoid the spurious oscillations around the interface or other discontinuities and shocks. To overcome these drawbacks, various high-resolution, non-linear techniques have been developed, often using flux/slope limiters. Possible schemes are, for example, the MUSCL scheme (van Leer [165]), HRIC or CICSAM (Ubbink and R.I.Issa [162]). Comparisons of HRIC and CICSAM methods have been discussed by Waclawczyk and Koronowiczy [167, 168].

The snapshot of a typical algebraic VOF model is given in Fig. 6.2a. The simulation has been run with OpenFOAM. The diffusive interface clearly can be observed, as the droplets which are generated in the surface break-up are smeared.

Advection-Diffusion (LB) models

Aside from numerical diffusion, which is introduced by the insufficient discretization of the gradient operator, a minimum interface diffusion can also be introduced intentionally by using advection-



(a) OpenFOAM, algebraic model

(b) Own implementation, LB advection-diffusion model

Figure 6.2: Different VOF methods

diffusion models at low diffusivities. A generalized advection-diffusion-equation reads:

$$\frac{\partial \Phi}{\partial t} + \mathbf{v}\nabla\Phi - \chi\Delta\Phi = q \tag{6.5}$$

for a diffusive quantity Φ, with external source terms q and (constant) diffusivity χ. In the limit of zero diffusivity, the equation corresponds to the advection equation (Eq. 6.4). The diffusivity χ, is not allowed to be set to zero directly, because this would change the type of the PDE back to hyperbolic, having great impact on the chosen discretization scheme. A minimum numerical diffusivity has to be introduced to guarantee stable and meaningful simulations, which as a price for that, leads to a smearing of the interface. Exemplarily, snapshots of a dam break simulation with an LB advection-diffusion model are given in Fig. 6.2b. The interface smearing clearly can be seen, which is not appropriate to simulating macroscopic free surface flow problems with sharp interfaces.

Geometry-based discretization of the advection equation

In geometric discretizations of the advection equation, in contrast to the aforementioned methods, the exact position of the free surface in the interface cells is determined in a reconstruction step. Subsequently, either a Eulerian, flux-based advection or a Lagrangian advection of the surface takes place during the advection step. Note that the transport equation is evaluated for interface cells only. An inherently sharp interface with a width of at most one grid cell is reconstructed.

The first VOF algorithm can be traced back to the SLIC (Single-Line-Interface- Calculation) algorithm of Noh and Woodward [125], which used a constant surface reconstruction scheme and operator splitting for the advection step. Hirt and Nichols [83] introduced the SOLA-VOF algorithm with a piecewise linear interface reconstruction (PLIC) for the treatment of the free-surface boundary conditions. The piecewise linear concept was adapted to the surface reconstruction step in the work of Youngs [178], which also represented the surface as a set of linear plane segments. Since then, several developments in the field of VOF methods have appeared. Among the surface reconstruction schemes, Puckett [133] and Pilliod [130] present second-order schemes for the evaluation of the surface normal from the discrete fill level information. For the advection step, unsplit methods (Pilliod and Puckett [131]) allow to treat all advection directions at once but are still based on a PLIC surface

reconstruction. Zhang and Liu [181] recently presented the polygonal area mapping (PAM) method, which represents the interface as piecewise polygons. They trace points on the polygon boundaries along and evaluate the fluxes via polygon-clippings. The results in 2D look very promising, yet the extension of the algorithm to 3D non-uniform grids seems to be computationally expensive. On top of SLIC and PLIC methods, it is also possible to use higher order interpolation methods in order to obtain a smooth free surface. Ginzburg and Wittum [54] propose a cubic spline interpolation of the free surface on a staggered finite volume grid. Surface points on the reconstructed PLIC interface are used to reconstruct a smooth interface layer without jumps or kinks. The extensions of this two-dimensional reconstruction step to 3D seems to be quite demanding and computationally expensive.

Mesoscopic discretization of the advection equation

The previously described Lattice Boltzmann advection scheme (section 5.4) can be considered as a special, geometry-based VOF method with a mesoscopic advection model. The flux terms between neighboring cells are expressed in terms of particle distribution functions:

$$\Phi_i = \varepsilon_i \cdot \left[f_i^t - f_I^{t+1} \right] \tag{6.6}$$

with the two antiparallel particle distribution functions $f_{i,I}$ entering or leaving the corresponding cell and the face fill levels ε_i that are calculated on the basis of a simplified surface reconstruction, see Eq. 4.22.

6.2 Details on the implementation of a PLIC method

For the basis of our implementation we use a geometric discretization of the advection equation, in combination with a PLIC method for the interface reconstruction. The use of a method with a sharp interface representation is essential for the coupling to a pure free surface model. As the free surface boundary is represented via a pressure boundary condition at the interface, a sharp interface of a width of at most one cell is needed. Otherwise, the location of the flow boundary condition would not be exactly clear. The diffusive interface approach is more suitable for multiphase flow codes, where the diffusive interface region can be represented via a blending function between the two fluid properties and in the momentum equations of both phases. On top, free surface flow simulations on a small scale demand diffusive interfaces and the simulation of the second phase, as this corresponds to the real interface physics. In nature, the thickness of air-water interfaces is way below the grid resolution, so that for our large-scale simulations, the use of a diffusive interface model is out of question.

In a weakly compressible CFD approach, as the LBM, the VOF fill level ε is *not* conserved, so that a recourse to the continuity equation and the principle of conservation of mass is used to derive the advection algorithm:

$$\frac{D\rho}{Dt} = \frac{\partial \rho}{\partial t} + \nabla(\mathbf{v} \cdot \rho) = 0 \tag{6.7}$$

We discretize this equation with a classical finite volume method by integrating the equation over the control volume and applying the divergence theorem to obtain a surface integral for the convective term:

$$\int_\Omega \frac{\partial \rho}{\partial t} d\Omega + \int_\Omega \nabla \cdot (\mathbf{v}\rho) d\Omega = \frac{\partial}{\partial t} \int_\Omega \rho \; d\Omega + \int_\Gamma (\mathbf{v}\rho) \cdot \mathbf{n} d\Gamma \tag{6.8}$$

where $\mathbf{n}$ is the outward pointing normal vector on the corresponding face of the control volume. The first integral over the system volume yields - according to the definition of the intensive density ρ - the mass in one control volume, as corresponding extensive system property. For the evaluation of the surface integral, several different approaches may be taken, which will be discussed later in detail. So far, the flux term Φ_i is introduced, denoting the flux on the i-th face of the control volume, so that Eq. 6.8 yields

$$\frac{\partial m}{\partial t} + \sum_i \Phi_i = 0 \tag{6.9}$$

Discretizing in time with an explicit Euler finite difference scheme leads to

$$m^{t+1} = m^t - \sum_i \Phi_i \cdot \Delta t \tag{6.10}$$

For the evaluation of the flux terms, two steps are mandatory in geometry-based VOF methods: at first, the surface has to be reconstructed out of the fill level information. Then, the flux is calculated and the new fill level of an interface cell hence is calculated via

$$\varepsilon^{t+1} = \frac{m^{t+1}}{\rho^{t+1}} = \frac{m^t - \sum_i \Phi_i \cdot \Delta t}{\rho^{t+1}} = \frac{\varepsilon^t \rho^t - \sum_i \Phi_i \cdot \Delta t}{\rho^{t+1}}. \tag{6.11}$$

for an *unsplit method*, where all flux directions are treated at once. In a split method, each advection direction is treated separately. Exemplarily, for a three-dimensional advection problem with 6 advection directions, this leads to two intermediate values ε^* and ε^{**} for the fill level:

$$\varepsilon^* = \frac{\varepsilon^t \rho^t - \Phi_0 - \Phi_1}{\rho^{t+1}}, \; \varepsilon^{**} = \frac{\varepsilon^* \rho^t - \Phi_2 - \Phi_3}{\rho^{t+1}}, \quad \text{and} \quad \varepsilon^{t+1} = \frac{\varepsilon^{**} \rho^t - \Phi_4 - \Phi_5}{\rho^{t+1}} \tag{6.12}$$

Between the evaluation of the fluxes, intermediate surface reconstruction steps take place. In order to avoid anisotropic effects, the order of directions may be swept every time step. The models for surface reconstruction and flux calculation, which are finally yielding the flux terms Φ_i will be discussed in the following.

6.2.1 Surface reconstruction scheme

Two different surface reconstruction schemes, namely the simplified fill level averaging which was used in the LB-based mesoscopic method, and a common PLIC interface reconstruction scheme are compared in this work. In the *averaged cell fill level approach*, the moistened area between two cells is determined without geometrical surface reconstruction by averaging the fill level of the two cells between which the mass exchange takes place. Körner et al. [95] use such a surface reconstruction scheme for the LB free surface model, which has already been discussed in section 5.4. For adjacent

(completely filled) fluid cells, the whole edge is moistened, so that if either of the two participating cells is a fluid cell, the moistened area is set to one, so that the final formulation reads

$$\varepsilon_i = \begin{cases} 1.0 & \varepsilon\left(\mathbf{x}+\mathbf{e}_i\right) = 1.0 \\ 0.0 & \varepsilon\left(\mathbf{x}+\mathbf{e}_i\right) = 0.0 \\ \frac{\varepsilon(\mathbf{x}+\mathbf{e}_i)-\varepsilon(\mathbf{x})}{2} & \text{else} \end{cases} \tag{6.13}$$

This simple scheme leads to good results with minimum computational effort, as no detailed surface reconstruction needs to be done, and even the normal vector information is not necessary. The global shape and advection behavior is represented fairly accurately, although oscillations can still occur on the free surface.

Piecewise linear interface reconstruction (PLIC)

Alternatively, a piecewise linear interface reconstruction method (PLIC method) can be used [178, 66]. This type of method determines the free surface orientation more accurately and avoids oscillations on the interface. The free surface is represented as a line segment (in 2D) or a plane (in 3D), which can be uniquely described by its unit normal vector $\mathbf{n} = (n_1, n_2, n_3)^T$ and the distance α to a point of origin:

$$\mathbf{x} \cdot \mathbf{n} = \alpha \tag{6.14}$$

The geometric base configuration is depicted in Fig. 6.3. The expression for the volume V below a plane for a unit cell with $\Delta x_i = 1.0$ yields

$$V(\mathbf{n}, \alpha) = \frac{1}{2n_1n_2n_3}\left[\alpha^3 - \sum_{j=1}^{3} H(\alpha - n_j)\left(\alpha - n_j\right)^3 + \sum_{j=1}^{3} H(\alpha - \alpha_{max} + n_j)\left(\alpha - \alpha_{max} + n_j\right)^3\right] \tag{6.15}$$

with Heaviside function H, $\alpha_{max} = \sum_{j=1}^{3} n_j$ and a plane parameter α. The cut volume of the reconstructed interface plane and the unit cell is known in advance and given by the cell fill level ε. Before the flux terms can be computed, the surface normal has to be evaluated and the plane constant α, describing the distance of the plane to the cell origin, has to be determined.

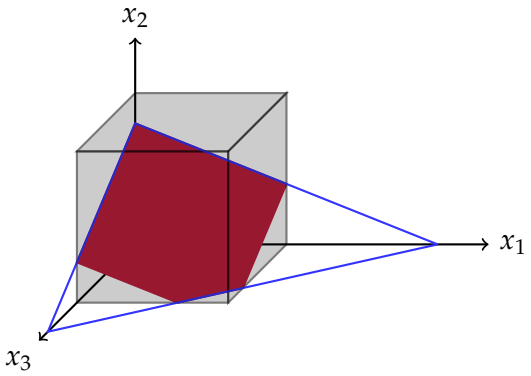


Figure 6.3: PLIC base configuration

Calculation of the surface normal

The normal vector is determined by a discrete approximation as, for example, by the normalized gradient of the fluid fraction variable ε:

$$\mathbf{n} = -\frac{\boldsymbol{\nabla}\varepsilon}{\|\boldsymbol{\nabla}\varepsilon\|}. \tag{6.16}$$

The gradient is obtained from the surrounding cell fill levels:

$$\boldsymbol{\nabla}\varepsilon = \frac{1}{2\Delta x}\begin{pmatrix} \bar{\varepsilon}_x(\mathbf{x}+\mathbf{e}_0) - \bar{\varepsilon}_x(\mathbf{x}+\mathbf{e}_1) \\ \bar{\varepsilon}_y(\mathbf{x}+\mathbf{e}_2) - \bar{\varepsilon}_y(\mathbf{x}+\mathbf{e}_3) \\ \bar{\varepsilon}_z(\mathbf{x}+\mathbf{e}_4) - \bar{\varepsilon}_z(\mathbf{x}+\mathbf{e}_5) \end{pmatrix} \tag{6.17}$$

where $\bar{\varepsilon}_{x,y,z}(\mathbf{x})$ are averaged values of the neighboring cells [129]:

$$\bar{\varepsilon}_x\,(x,y,z) = \sum_{i=-1}^{1}\sum_{j=-1}^{1} \varepsilon(x\ , y+i, z+j)\cdot w_{i,j} \tag{6.18}$$

$$\bar{\varepsilon}_y\,(x,y,z) = \sum_{i=-1}^{1}\sum_{j=-1}^{1} \varepsilon(x+i, y\ , z+j)\cdot w_{i,j} \tag{6.19}$$

$$\bar{\varepsilon}_z\,(x,y,z) = \sum_{i=-1}^{1}\sum_{j=-1}^{1} \varepsilon(x+i, y+j, z\ )\cdot w_{i,j} \tag{6.20}$$

Different weighting factors $w_{i,j}$ can be used. The weight $w_{i,j} = 1$ corresponds to the center-of-mass approach of Puckett and Saltzman [132], but according to Parker and Youngs [129], the following weights produce the best results:

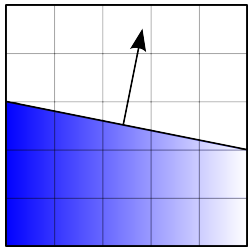
$$w_{0,0} = 4, \quad w_{\pm1,0} = w_{0,\pm1} = 2 \quad \text{and} \quad w_{\pm1,\pm1} = 1 \tag{6.21}$$

These weighting factors correspond to the weighting factors of finite difference approximations of first-order derivatives. Because the fill level in a cell is limited to the range $\varepsilon \in [0.0, 1.0]$, it is clear that the resulting normal vector is only an approximation. It can be shown that the approximation is between first- and second-order accurate. In addition, enhanced schemes have been developed to

improve the accuracy of normal vectors. The least-squares VOF interface reconstruction algorithm (LVIRA, [133]) uses an iterative procedure to obtain surface normals with an accuracy of $\mathcal{O}(\Delta x^2)$. The efficient LVIRA (ELVIRA, [130]) accelerates the iterative process by selecting several candidates for the surface normal, then choosing the one which minimizes a predefined error norm. The work of Miller and Colella [116] extends this idea to three dimensions and determines 72 to 144 candidates for the normal vector. Each candidate requires a surface reconstruction, 27 intersections of the reconstructed plane and grid cells, and the evaluation of the error norm, which leads to high computational costs. To lower these costs, we implement a reduced version, where only 27 candidates are tested; specifically, the central, forward, and backward finite differences are evaluated for each component of the normal vector. All 27 possible combinations are checked in a manner similar to that used in the ELVIRA method. The quality of the calculated normal vector is examined with a straightforward test setup, depicted in test case setup 6.1. An inclined surface with normal vector $\mathbf{n}$ is mapped to the grid. The normal vector is then numerically calculated out of the cell fill levels and compared to the theoretical target value.

Parameter	Value
domain	$[0,10]^3$
grid	6^3
α	5.5
Φ_x	$0..\frac{\pi}{2}$
Φ_y	0.0

(a) Initial setup



(b) Geometry

Test case 6.1: Normal vector benchmark

In Fig. 6.4, the L1 error norm for four different approaches is given for a plane with an inclination from 0 to $\pi/2$. The use of Parker-Youngs weighting factors reduces the error by approximately a factor of two. The modified ELVIRA correction further diminishes the error by another approximate order of magnitude.

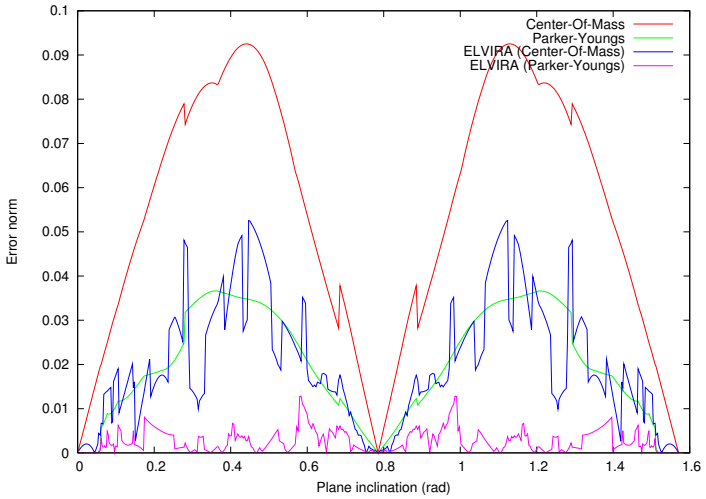


Figure 6.4: Error norm for the normal vector calculation of an inclined plane, for different angles and normal vector calculation schemes

Calculation of the interface constant alpha

Subsequently, the only remaining unknown value for the linear surface reconstruction (Eq. 6.14) is the distance α between the surface plane and a coordinate origin. Dealing with Eq. 6.15 is demanding, as the six Heaviside functions depend on the components of the surface normal in the interface cell. The geometric reason for the occurrence of these correction terms is illustrated in Fig. 6.5. The intersection of a plane with a given surface normal is shown for increasing values of α. The nature of Eq. 6.15 is changed as soon as a vortex of the control volume is reached by the surface. The corresponding volumes are often referred to as *critical* volumes, and each Heaviside term corresponds to one unique critical volume. Solving analytically for α as an inverse of this expression is not generally possible; therefore, in some volume regions, *Brent's method* [14] is used to determine α iteratively. However, the position of the plane can then be uniquely determined. Once the plane parameter α is known, the flux calculation is straightforward. Details on critical volumes, analytical solutions for some volume regions and a descriptive explanation of a 2D-PLIC is given in Appendix B.

6.2.2 Flux calculation

Once the surface has been reconstructed, the evaluation of the flux terms Φ_i is straightforward, no matter which reconstruction technique has been chosen. We show a time-constant and time-linear case, which has shown to be sufficient for our applications. In general, higher-order time integration is possible.

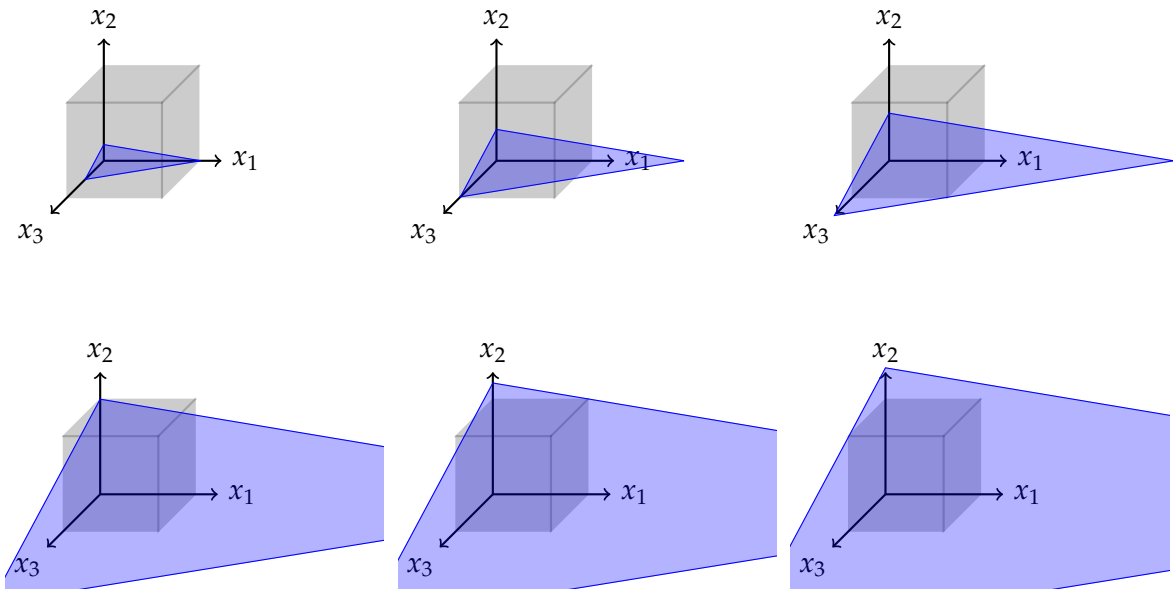


Figure 6.5: PLIC interface construction: six critical volumes for a plane with $\mathbf{n} = (0.156, 0.935, 0.313)$

Constant in time

In a straightforward approximation for the fluxes Φ_i, the moistened portion of the control volume surface (*face fill level* ε_i^t) is assumed to be constant during the advection step:

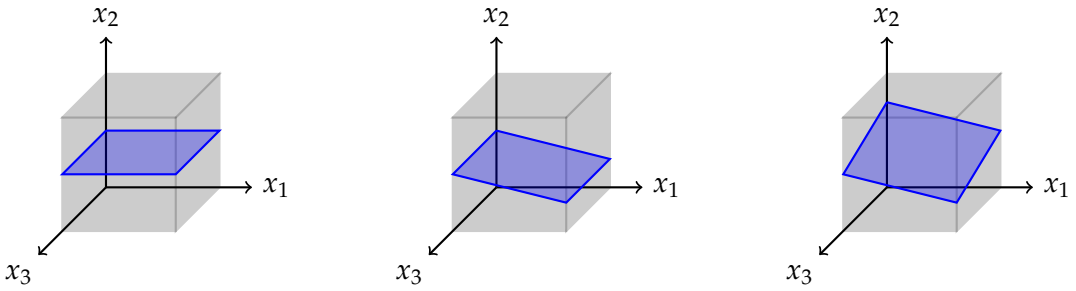
$$\Phi_i = (\mathbf{v} \cdot \hat{\mathbf{n}}\rho)_i^t \cdot \varepsilon_i^t \cdot \Delta t \tag{6.22}$$

Hence, for the flux calculation, the face fill level ε_i^t has to be evaluated.

For the simple approach, this is done by evaluating Eq. 6.13. The face fill levels are determined locally for every interface cell and the mass balance is evaluated directly on-the-fly. Face fill levels for one cube face are the same by definition, no matter which of the two participating cells evaluates the value.

For the PLIC reconstruction, three basic cases have to be distinguished, once the PLIC plane is reconstructed, see Fig. 6.6. If two components of the normal vector ($n_2 = n_3 \approx 0.0$) are zero, the surface is axis-parallel. One face is completely wet resp. dry, and all four other faces have a face fill level which equals the total cell fill level, $\varepsilon_i = \varepsilon$. In case of one zero component of the surface normal ($n_3 \approx 0.0$), a reduced two-dimensional PLIC algorithm can be used to determine the moistened area. For the most complex case, when all components of the normal vector are nonzero, an unmodified PLIC algorithm is needed.

Note that the simplifications mentioned above are not only reducing computational time but also are mandatory, as - if these special cases were not treated carefully enough - the evaluation of Eq. 6.15 would fail, as normal vector components appear in the denominator of the overall expression.



(a) $n_2 = n_3 \approx 0.0$ (b) $n_3 \approx 0.0$ (c) $n_i \neq 0.0$

Figure 6.6: Three basic PLIC cases

Linear in time

In a time-linear advection step, the deformation of the free surface during the advection step is considered. After the surface reconstruction and evaluation of face cell velocities, the characteristics are traced back and the flux is calculated as the intersection volume of cuboids and the reconstructed surface segment. The fluxes Φ_i are determined using the intersection of the fluid domain and the fraction of the cell volume that is advected, for example, $V_{1,cell} = (v_1 \Delta t)\Delta x_2 \Delta x_3$ in the x_1-direction.

In contrast to the approach for a constant fill level in time described in section 6.2.2, a time-linear determination of the fluxed volume does not introduce excessive mass:

$$\Phi_i = (\mathbf{v} \cdot \hat{\mathbf{n}}\rho)_i^t \cdot \int_t \varepsilon_i^t dt \tag{6.23}$$

By construction, the maximum fluxed volume can never be exceeded.

We use time-linear advection exclusively with the PLIC surface reconstruction. The flux calculation requires the intersection of the reconstructed PLIC plane segment with arbitrary cuboidal volumes. If the cuboid includes the point of origin in the permuted cell configuration, the evaluation of the volume is straightforward. In all other cases, a new point of origin has to be defined, leading to a new modified value for $\tilde{\alpha}$, which is easily obtained as

$$\tilde{\alpha} = \alpha - \mathbf{n} \cdot \mathbf{x} = \alpha - n_1 \cdot x_1 - n_2 \cdot x_2 - n_3 \cdot x_3 \tag{6.24}$$

where $\mathbf{x}$ refers to the new point of origin and $\mathbf{n}$ refers to the permuted unit normal vector on the PLIC. The cut volume of a plane and an arbitrary cuboidal control volume with extents Δx_i is given by

$$V(\mathbf{n}, \alpha, \Delta x_i) = \frac{1}{2 n_1 n_2 n_3} \left[\alpha^3 - \sum_{j=1}^{3} H(\alpha - \Delta x_j n_j) \left(\alpha - \Delta x_j n_j\right)^3 + \sum_{j=1}^{3} H(\alpha - \tilde{\alpha}_{max} + \Delta x_j n_j) \left(\alpha - \tilde{\alpha}_{max} + \Delta x_j n_j\right)^3 \right] \tag{6.25}$$

with $\tilde{\alpha}_{max} = \Delta \mathbf{x} \cdot \mathbf{n} = \sum_{j=1}^{3} \Delta x_j n_j$. This volume V is evaluated for the reconstructed linear surface (normal vector $\mathbf{n}$, plane parameter α) and the portion of the cell volume, which is advected to the neighboring cell, $\mathbf{v}_i \Delta t$: $\Phi_i = V(\mathbf{n}, \alpha, \mathbf{v}_i \Delta t)$ for an LB unit cell with $\Delta x_i = 1$.

6.2.3 Excessive mass

Depending on the type of flux calculation scheme, the need for mass distribution arises. In time constant flux calculation methods, the time evolution of the free surface is not considered, which might result in a flux larger than the total amount of fluid inside the donor cell, and a negative fill level. Secondly, the neighboring acceptor cells receive too much mass. The same situation occurs when a cell is filled up with fluid. The resulting fill level is larger than one, while in this case, the neighboring cell does not receive enough (i.e. in most of the cases 0.0) mass. Hence, the excessive (positive or negative) mass m_{ex} has to be distributed among neighboring interface cells. Different algorithms are possible - a velocity-based distribution leads to best results:

$$\Delta m_i = m_{ex} \cdot \frac{\mathbf{e}_i\mathbf{v} \cdot H(\mathbf{e}_i\mathbf{v})}{\sum_j \mathbf{e}_j\mathbf{v} \cdot H(\mathbf{e}_j\mathbf{v})} \quad \forall \text{ neighboring interface cells } i \tag{6.26}$$

where $\mathbf{v}$ is the average cell velocity in the cell which changed its state, and Heaviside function H.

In time linear flux calculation approaches, this is not necessarily needed. For emptying interface cells, the time linear flux evaluation guarantees that the total flux never exceeds the physical maximum value. Only new interface cells, which have been fluid cells before and hence skip the interface update (as both old and new fill level should yield 1.0), have to be initialized with meaningful values. In a time linear approach, they simply can re-do the previous update interface step. The same holds for cells next to filled-up interface cells. The neighboring interface cells can provide flux information on the basis of cell-centered, extrapolated velocities, after the update-interface step.

This methodology strongly supports the use of time linear flux calculation, as the mass distribution is a lot more natural and physical, compared to the heuristic velocity-based approaches of time constant methods (Eq. 6.26).

6.2.4 Grid refinement

At the transition between grids of two different refinement levels, a local surface reconstruction is necessary (Fig. 6.7) for the data transfer from a coarse to a fine grid. The surface is reconstructed in a cell on the coarse grid using the PLIC algorithm. On the basis of the resulting unique half-space with normal vector $\mathbf{n}$ and plane parameter α, the corresponding eight cells on the fine grid are specified. The fill levels are obtained using the intersections of the half-space and the cell cuboids (Eq. 6.25 with $\Delta x_i = \Delta x_{i,fine} = 0.5\Delta x_{i,coarse}$, and, if applicable, Eq. 6.24). The nodal states are set by a point-in-object test, which determines if the nodes on the fine grid are inside or outside the fluid volume. The resulting set of fine cells is transferred to the fine block and serves as a boundary condition in the advection step. In the opposite direction (fine to coarse), the cell information is coarsened on the fine block in a straightforward way by a simple summation, and then transferred to the coarse grid.

This algorithm so far has been used for a priori refinement of initialized free surface domains and at the interface of grids of different refinement levels during advection test cases.

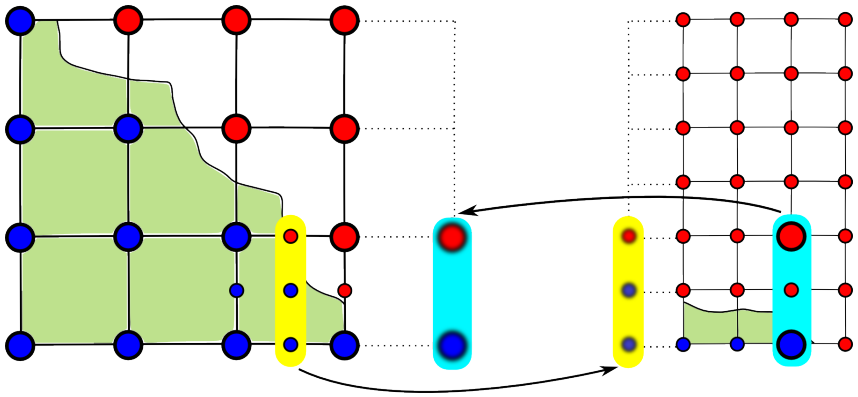


Figure 6.7: Non-uniform block transition with ghost layers (dotted lines)

6.2.5 Calculation of the interface curvature

Surface tension effects have not been included in this work, as they do not play an important role for problems on the considered macroscopic length scale. Importance of surface tension grows for small scale problems, where e.g. the influence of droplets or the wetting behavior of the fluid highly influence the global fluid behavior. Surface tension effects are related to the interface curvature. As an outlook for future work, a short overview over the curvature calculation algorithms in literature will be given, together with a brief presentation of two methods which have shown to be most promising for our hybrid VOF-LBM approach.

In two-dimensional VOF codes, curvature estimation by means of local, generalized height-functions has shown to be very successful. Depending on the interface orientation, the water height is locally reconstructed, so that in a second step, the second derivative (i.e. the curvature) can be calculated. Some groups extend this idea to three space dimensions, which finally leads to complex geometrical operations, see e.g. Wemmenhove [173] and the work of the MARIN group.

Recently, the least squares fitting approach of Meier et al. [115] introduced a variable reduction for the curvature calculation in two-dimensional VOF methods. Instead of using a classical finite difference stencil, three artificial parameters Σ, Ξ and Ω are introduced:

$$\Sigma = \sum_{i=-1..1} \sum_{j=-1..1} \varepsilon_{i,j} - \varepsilon_{0,0} \tag{6.27}$$

$$\Xi = \varepsilon_{0,0} \tag{6.28}$$

$$\Omega = \left(\frac{\min(|n_x|, |n_y|)}{\max(|n_x|, |n_y|)} \right) \tag{6.29}$$

The parameter Σ is related to the variance of fill levels in the proximity of the interface cell under consideration, Ξ is the fill level of the cell itself, and Ω qualifies the surface inclination by means of the surface normal components n_x and n_y. On the basis of these parameters, a polynomial curvature estimator is defined, which is fitted to reference data by a least-squares approach. Initial 2D benchmarks have been successfully run by the author, so that the extension of this approach to 3D should be considered in future work.

Last, but not least, a fallback to level-set methods for the curvature calculation seems to be promising. Several hybrid VOF-LS methods already adapted this idea to use both methods at the same time,

and mutual correction steps to get consistent results. Hence, the level-set signed distance function in the vicinity of the phase interface has to be reconstructed out of VOF fill level information. On the basis of a surface reconstruction, as e.g. obtained by the PLIC, these distances can be evaluated. Initial validations have been done (Fig. 6.8), but as curvature driven flows and capillary waves are not in the focus of the work, this short outlook ends the curvature-related discussions.

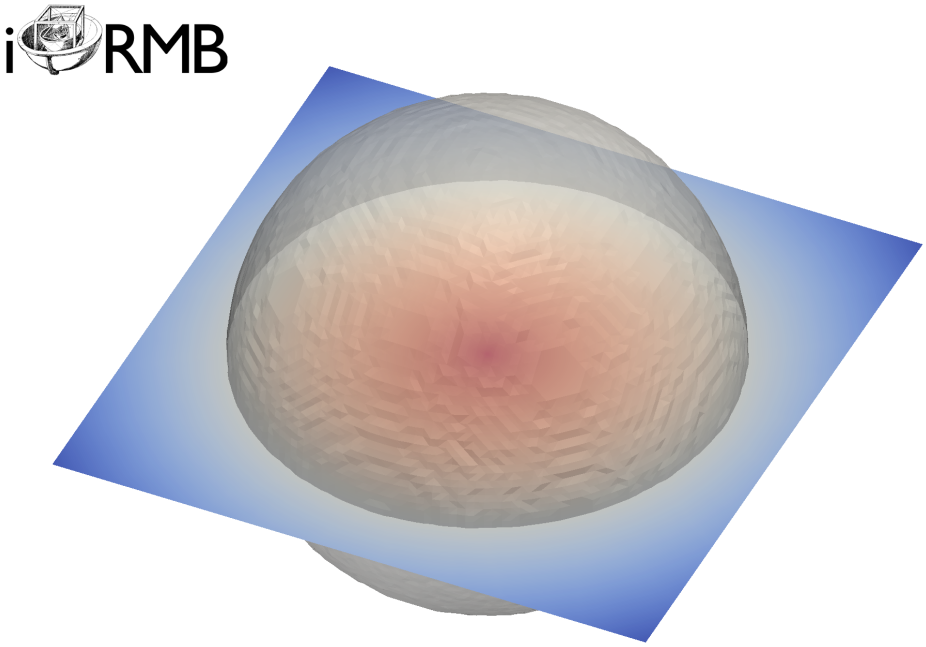


Figure 6.8: Isosurface on zero-level set after reconstruction from VOF data

6.3 Verification

The quality and the convergence behavior of the new algorithm is first checked with pure advection test cases. The LB collision and propagation steps are omitted. Instead, the equilibrium distribution functions for a given velocity field are prescribed according to a velocity potential $\Psi(x,y)$ as

$$\mathbf{u} = \boldsymbol{\nabla} \times \boldsymbol{\Lambda} \tag{6.30}$$

with $\mathbf{u} = (u,v,0)$ and $\boldsymbol{\Lambda} = (0,0,\Psi(x,y))$. The error is quantified using the relative distortion of the material area E_r, i.e., the L1-norm of the difference between actual and target fill levels:

$$E_r = \frac{\sum_\Omega \left| \varepsilon^t_{act} - \varepsilon^t_{tg} \right|}{\sum_\Omega \varepsilon^t_{tg}} \tag{6.31}$$

Some basic test cases from the advection community serve to validate the advection scheme. Essentially, the overall conservation of shape is checked and compared to other state-of-the-art methods by L1 and L2 error norms. The final example of the advection of a sphere serves to estimate the convergence rate of the algorithm. Please note that the two-dimensional test cases are realized as quasi-2D equivalents with periodic boundary conditions in the third space direction. All simulations are carried out within the three-dimensional code framework VirtualFluids.

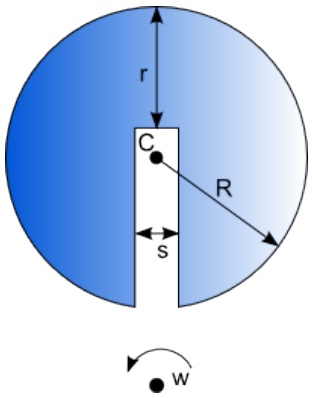
6.3.1 Notched circle

The notched circle setup with rotational velocity field (test case setup 6.2) has been introduced by Zalesak [180]. It shows good conservation of shape (Fig. 6.9) after one full rotation with the given stream function

$$\Psi(x,y) = -\frac{\omega}{2}\left[(x-x_0)^2 + (y-y_0)^2\right] \tag{6.32}$$

Parameter	Value
Domain	4m x 4m
slot	$r = 0.4m$, $s = 0.06m$
circle	$R = 0.5m$, $\mathcal{C} = (2.0m, 2.75m)$
origin	$\mathcal{O} = (2.0m, 2.0m)$
Lattice	$200 \times 200 \times 1$
ω (ω_{LB})	0.5 1/s ($0.00249 = 0.5 \cdot 0.00498$)
Δx (Δx_{LB})	0.02m (1.0)
Δt (Δt_{LB})	0.00498s (1.0)
T (T_{LB})	12.56952s (2524)
Courant C_r	0.25

(a) Parameters



(b) Geometry

Test case 6.2: Zalesak's notched circle

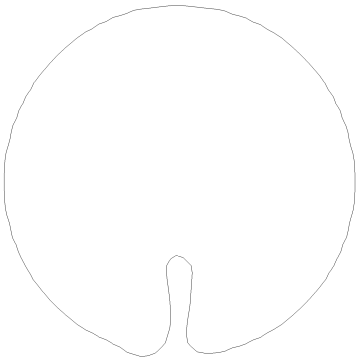


Figure 6.9: Notched circle after one rotation

In Tab. 6.1, the resulting errors are given for simulations with different surface reconstruction schemes (simple, PLIC time-constant, PLIC time-linear), normal vector schemes (CM/PY, both with and without ELVIRA), and split and unsplit methods. The split method in combination with the time-linear PLIC produces the best results, nearly independent of the method used for normal vector estimation. Our results are compared to those from other similar VOF methods as tested by Rudman [138]: a simple line interface reconstruction (SLIC, Noh and Woodward [125]), an SLIC approach with consideration to normal vector and surface orientation (Hirt and Nichols [83]), a VOF approach with flux correction (FCT-VOF, Rudman [138]), and a PLIC method (Youngs [178]). With a minimum error of $E_{r,min} = 0.0438$, our results are comparable to those from the methods tested by Rudman [138].

Normal vector	Time disc.	ELVIRA	Simple Approach	PLIC	
			time-constant	time-constant	time-linear
CM	unsplit	no	0.1252	0.1424	0.1356
CM	unsplit	yes		0.1305	0.1059
PY	unsplit	no		0.1369	0.1163
PY	unsplit	yes		0.1291	0.1079
CM	split	no	0.1328	0.1499	0.0485
CM	split	yes		0.1288	0.0442
PY	split	no		0.1332	0.0438
PY	split	yes		0.1270	0.0447
SLIC			0.0838		
Hirt-Nichols			0.0962		
FCT-VOF			0.0329		
Youngs			0.0109		

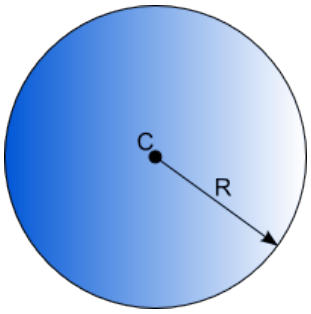
Table 6.1: Error norm for notched circle test case compared with results of Rudman [138]

6.3.2 Single vortex

Secondly, the Rider-Kothe single vortex test case [136] has been examined (test case setup 6.3). Here, the rotational velocity field is changing in time. After half of the rotation period, the velocity field is inverted, so that in the end, after one full advection period, the initial state should be restored.

Parameter	Value
domain	$[0, 1] \times [0, 1]$
grid	128×128
circle	$\mathcal{C} = (0.5, 0.75)$
Δx (Δx_{LB})	1/128 (1.0)
Δt (Δt_{LB})	0.000390625 (1.0)
n_{step}	5120
T (T_{LB})	2 (5120)
Courant	0.05

(a) Parameters



(b) Geometry

Test case 6.3: Rider-Kothe's single Vortex

The imposed velocity potential is given by

$$\Psi(x,y) = \frac{1}{\pi}\sin^2(\pi x)\sin^2(\pi y)2\cos(\pi t/T). \tag{6.33}$$

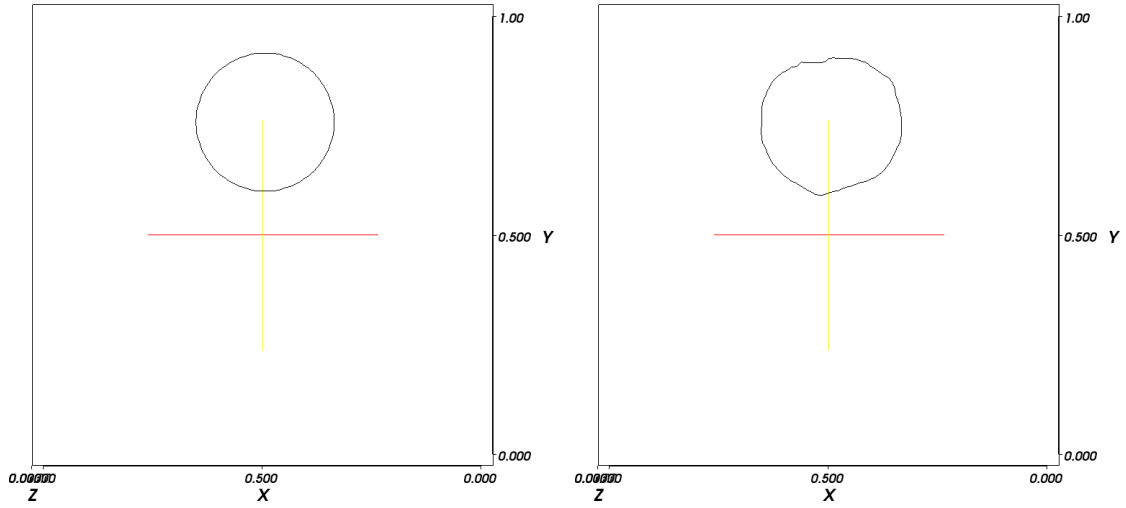
The initial setup and the test case parameters are given in test case setup 6.3. The material is stretched into a filament that spirals towards the vortex center. Following LeVeque [105], the additional cosine results in a flow problem that time-reverses (turning point $t = T/2$), so at time $t = T$ the initial setup should be restored. Our results are compared to those from Rider and Kothe [136]. For error norm E_g, the authors specify the absolute difference between target and actual fill levels

$$E_g = \sum_{\Omega} V_{cv} \left| \varepsilon^t_{act} - \varepsilon^t_{tg} \right| \tag{6.34}$$

which can be transformed to a relative error E_r norm by taking into account the total fluid volume of the advected body:

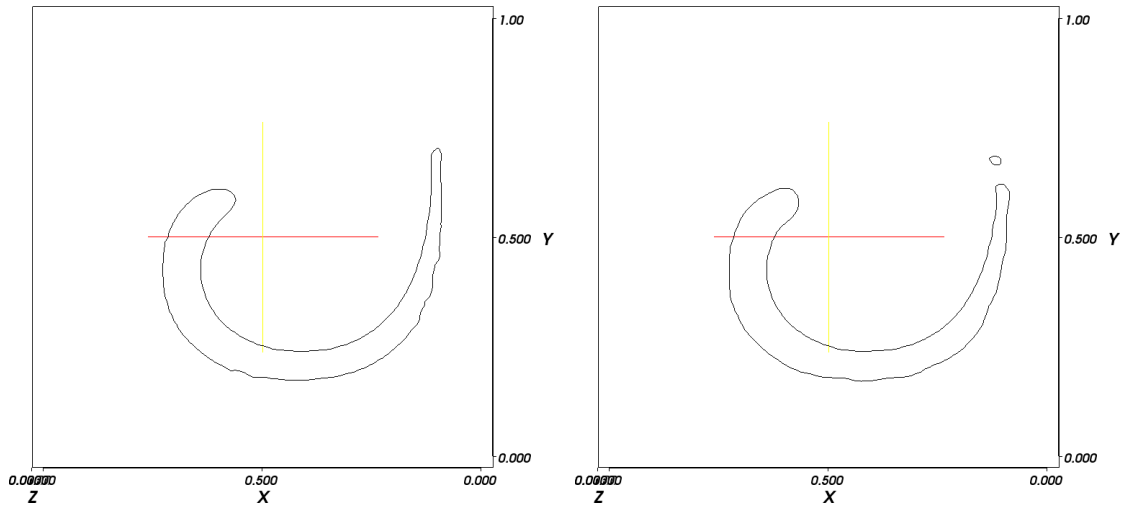
$$E_r = \frac{\sum_\Omega \left| \varepsilon^t_{act} - \varepsilon^t_{tg} \right|}{\sum_\Omega \varepsilon^t_{tg}} = \frac{\sum_\Omega V_{cv} \left| \varepsilon^t_{act} - \varepsilon^t_{tg} \right|}{\sum_\Omega V_{cv} \varepsilon^t_{tg}} = \frac{E_g}{V_{\text{obstacle}}} \tag{6.35}$$

For the Rider-Kothe circle with $V_{\text{obstacle}} = \pi r^2 = 0.07065$, we obtain the relation $E_r = 14.5 E_g$. In Tab. 6.2, the resulting relative errors E_r are given for various simulations. Again, the split method in combination with the time-linear PLIC produces the best results, with a minimum error $E_{r,min} = 0.0098$. This is comparable to the algorithms of Rider and Kothe [136] (2D PLIC with second-order normal vector determination and second-order time integration) and Harvie and Fletcher [77] (stream scheme, PLIC and unsplit time integration on the basis of streamlines and characteristics).



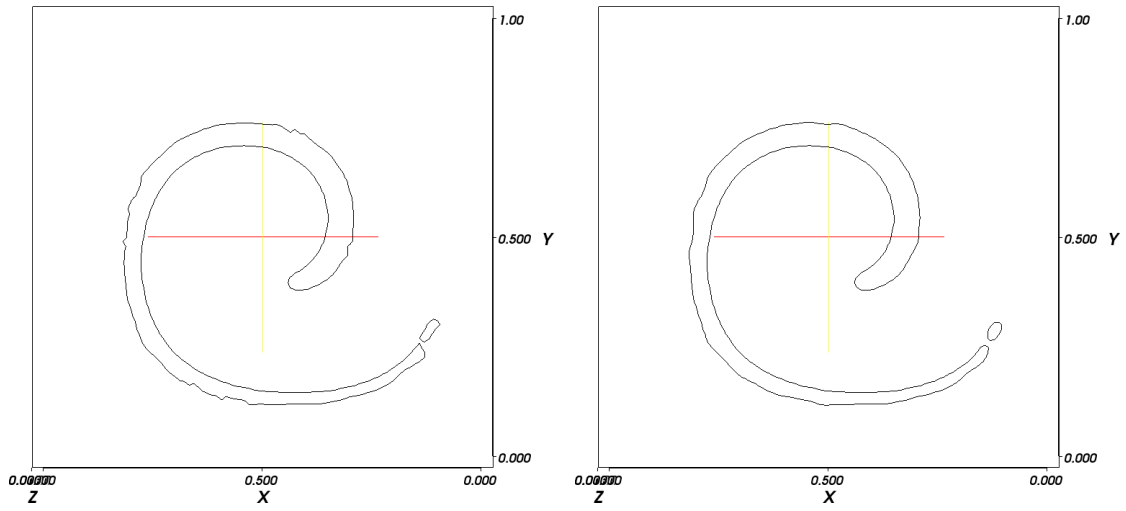
(a) t= 0

(b) t=1.0 T



(c) t=0.2 T

(d) t=0.8 T



(e) t=0.4 T

(f) t=0.6 T

Figure 6.10: Rider-Kothe reversed single vortex test case

Normal vector	Time disc.	ELVIRA	Simple Approach	PLIC	
			time-constant	time-constant	time-linear
CM	unsplit	no	0.056563	0.015114	0.018763
CM	unsplit	yes		0.011394	0.017291
PY	unsplit	no		0.010197	0.016721
PY	unsplit	yes		0.012050	0.017468
CM	split	no	0.054632	0.015242	0.013315
CM	split	yes		0.011029	0.009828
PY	split	no		0.010060	0.009932
PY	split	yes		0.010694	0.011435
Rider, Kothe [136]			0.001853		
Stream scheme [77]			0.003154		

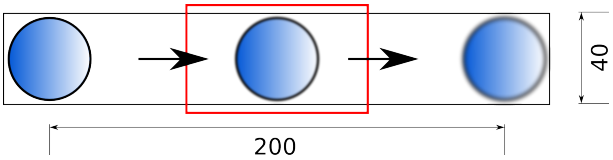
Table 6.2: Error norm for single vortex test case (T=2)

6.3.3 Convergence check

The order of convergence is validated with the orthogonal advection of a sphere of liquid (test case setup 6.4). The simulation is first run for different uniform grid resolutions from 5 to 40 nodes per sphere diameter. Secondly, a refined region (red box) with double resolution is introduced. The resulting grid is shown in Fig. 6.12 for the $120 \times 20 \times 20$ test case.

Parameter	Value
Lattice	$[0, 200] \times [0, 40]$
Sphere	$D = 36$

(a) Parameters



(b) Geometry

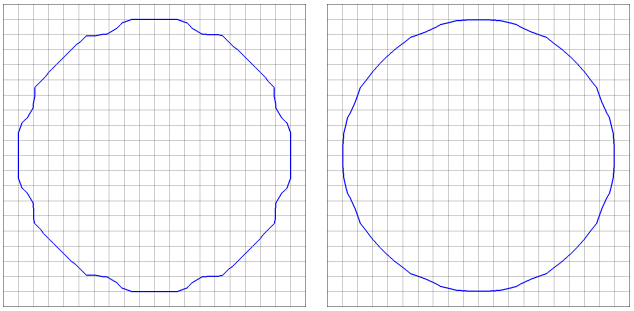
Test case 6.4: Pure advection

In recent work, we compared the actual position of the center of mass of the sphere x^t_{act} to the target value x^t_{tg} at discrete time steps t, as given in Janßen and Krafczyk [90]. We obtained an order of convergence of about 0.85 for the PLIC surface reconstruction and 1.2 to 1.6 for the simple averaging approach. It was initially surprising that the simple approach leads to higher rates of convergence; however, after comparing snapshots from the two simulations, it becomes clear that in the advection steps for the simple approach scheme, the sphere shape is degenerated to an artificial shape, whereas in the PLIC scheme, the sphere is advected nearly perfectly (Fig. 6.11). The distortion of the surface shape itself was not considered in the chosen center-of-mass-based error norm. Hence, in recent convergence checks, a full error norm on the basis of the actual and target fill levels ε_{act} resp. ε_{tg} has been defined according to:

$$E_1 = \frac{\sum_t \left| \varepsilon^t_{act} - \varepsilon^t_{tg} \right|}{\sum_t \left| \varepsilon^t_{tg} \right|} \tag{6.36}$$

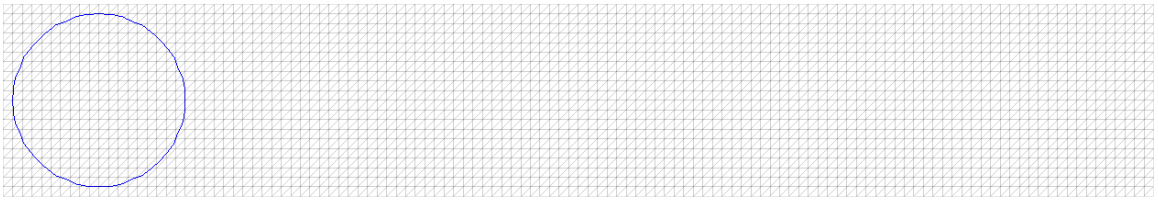
The resulting errors are dependent on both CFL number and grid spacing and are given in Fig. 6.13. An overall convergence order of around 1 and a linear dependancy on CFL number is noted. The low

order is due to the explicit time discretization of the interface advection step (which is first-order in time) and the surface reconstruction scheme (where the determination of the normal vector is between first- and second-order in space).

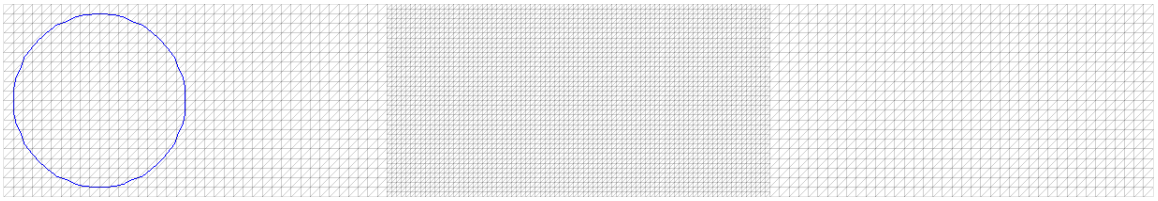


(a) Simple Approach (b) PLIC

Figure 6.11: Comparison of two surface reconstruction schemes



(a) Uniform



(b) Nonuniform

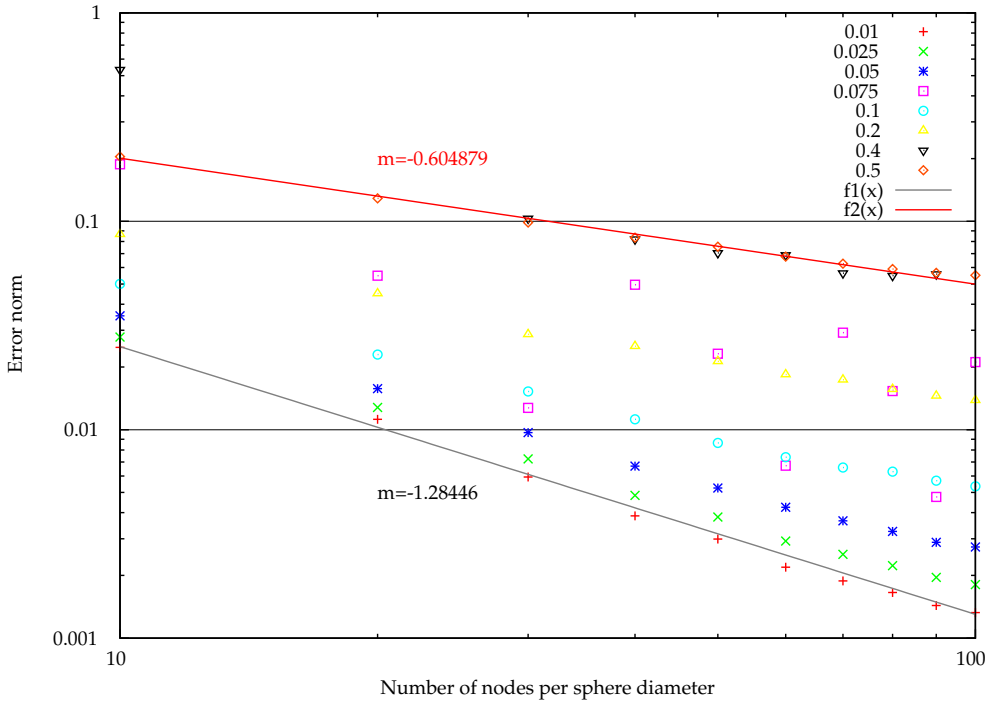
Figure 6.12: Initial grid layout (2D cut)

Grid refinement

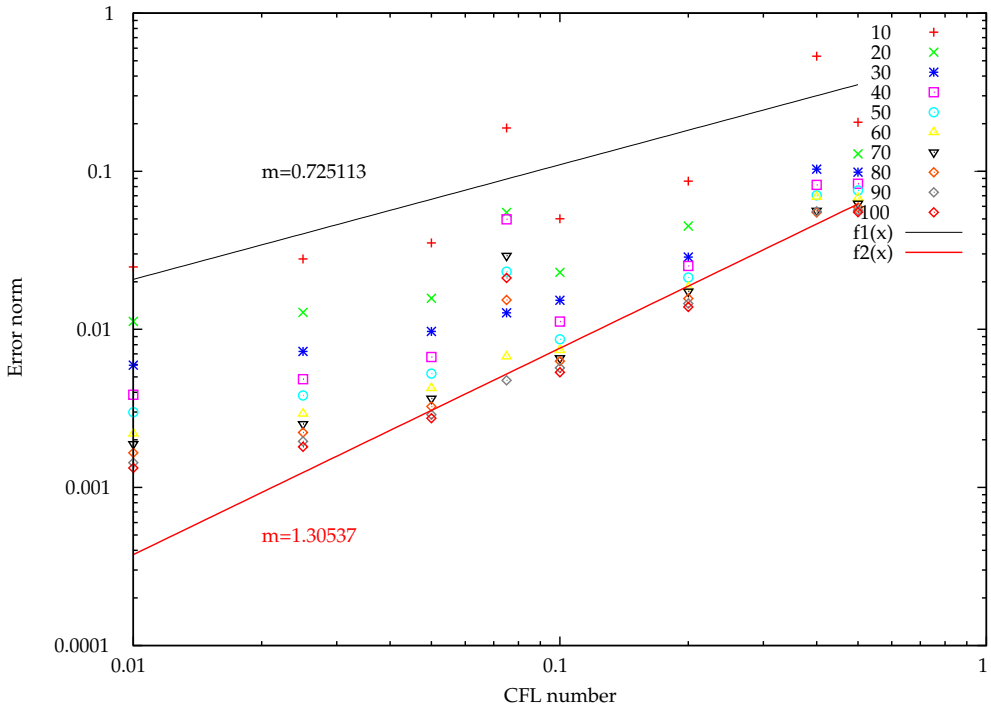
The non-uniform advection tests for the PLIC approach with a refinement region (Fig. 6.14b) lead to convergence rates similar to those from the uniform simulations (Fig. 6.14). We conclude that the chosen interpolation at the grid transition introduces no substantial negative effects.

Courant number	0.125	0.1	0.05	0.025
PLIC uniform	0.818	0.824	0.840	0.829
PLIC non-uniform	0.823	0.785	0.854	0.878

Table 6.3: Order of convergence for purely advective test case

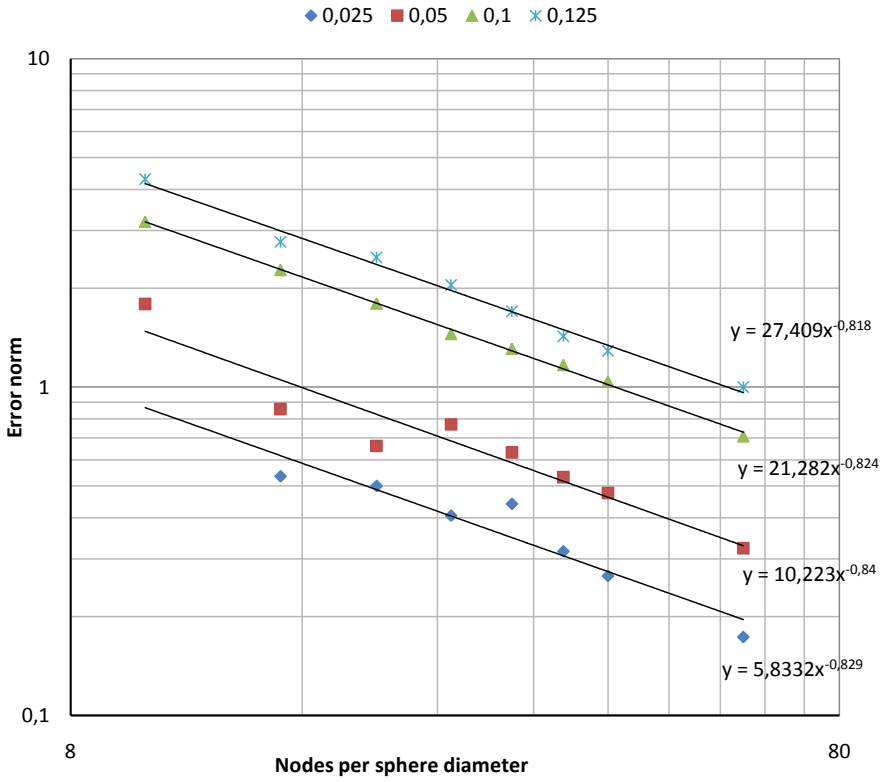


(a) Error vs. number of nodes for several advection velocities

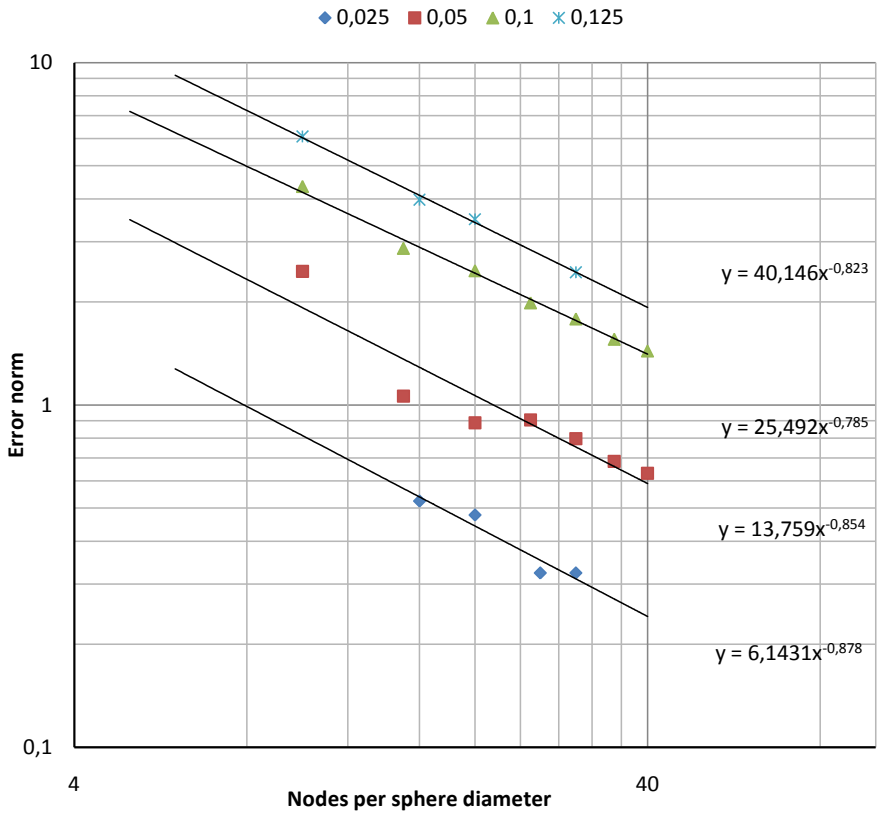


(b) Error vs. CFL number

Figure 6.13: Comparison of error norms for a purely advective test case, with different grid resolutions and CFL numbers



(a) uniform



(b) non-uniform

Figure 6.14: Comparison of error norms for a purely advective test case, with different grid resolutions and CFL numbers, with and without grid refinement

6.4 Conclusions

After the initial review of interface capturing and tracking methods, we proposed a Volume-Of-Fluid (VOF) interface capturing algorithm on the basis of piecewise linear, geometric interface reconstruction (PLIC). The surface normal is obtained from the discrete fill level information using a finite-difference approximation on the basis of Parker-Youngs weighting factors with a spatial accuracy of $\mathcal{O}(\Delta x)$ to $\mathcal{O}(\Delta x^2)$. For the time-discretization, an explicit Euler scheme is used, whereas the time-dependent change of the free surface position during the advection is considered in the geometrical flux calculation scheme.

The resulting algorithm has successfully been applied to standard advection test cases. An improved, iterative surface normal evaluation did not drastically improve the results, but degraded the overall performance.

In the following chapter, the advection algorithm will be coupled to a flow solver on the basis of the Lattice-Boltzmann method in order to simulate real- world free surface flow applications.

CHAPTER 7

Enhanced Free Surface Flow Model

In this chapter, we develop an enhanced, hybrid free surface flow model on the basis of the previously described advection scheme and the LBM. The LBM serves to calculate the flow field, which then advects the free surface. This vice versa leads to topological changes, which have to be transferred back to the LBM domain. In this chapter, the coupling of LBM and the advection scheme is presented and various validation examples and applications are given.

Concerning LB and free surface flow, several approaches have been taken. Ginzburg and Steiner [55] use a VOF-representation of the free surface and solve the corresponding transport equation with an additional LB model. The authors of [95, 156] use a VOF-scheme with flux- based advection on the basis of LBM distribution functions. Thürey and Rüde [155] compare level sets to the VOF approach. Those approaches do not lead to satisfying (i.e. mass conserving) algorithms in our existing non-uniform block-structured grid layout in VirtualFluids [41], which basically is the motivation of this work. In this work, a volume-of-fluid (VOF) method has been chosen to represent the interface. A sophisticated choice of the VOF control volume in the coupling with the LBM leads to a method which is especially suitable for a mass-conserving application on non-uniform block-structured grids and for fluid-structure interaction.

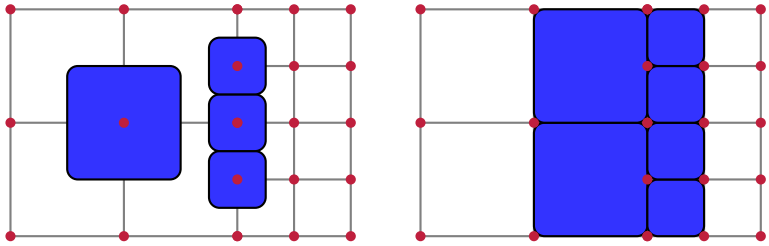
7.1 Hybrid LB-FV free surface algorithm

For a coupled, hybrid LB-FV-free surface algorithm both contributing solvers have to be extended and algorithms for the information exchange have to be specified. Topological information on the free surface location is available in the VOF context, whereas the information about density and velocity is provided in terms of particle distribution functions on the basis of LB lattice nodes.

The first, and most basic decision in the coupling of node- and cellbased solvers is the grid layout. The LBM provides solutions on equidistant, uniform grids. Pressure and velocities are - opposite to most Navier-Stokes solvers which act on staggered grids - available at the nodes directly. The finite

volume discretization of the free surface advection equation in combination with the PLIC surface reconstruction scheme asks for cell-centered values of fill-level and density, and face-centered values for advection velocities.

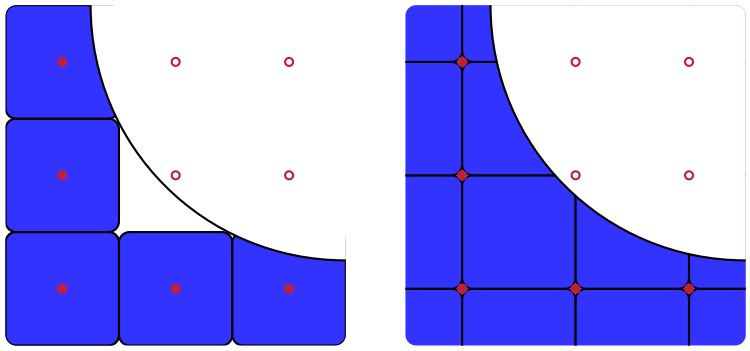
So far, the layout of the control volumes has not been specified, so that this is, however, a more or less free choice. In other free surface approaches the control volumes are directly assigned to the LB nodes ([95], Fig. 7.1a). In contrast to that, in a staggered grid layout, every cell (i.e. control volume) is spanned up by eight LB nodes (Fig. 7.1b). The advantage of the latter concept is the suitability for a mass-conserving application on non-uniform block-structured grids (Fig. 7.1). While in the node-based layout a gap occurs between the control volumes at the block transition, the cell-based layout provides direct contact between the control volumes of two different refinement levels so that the mass flux between coarse and fine blocks can be determined uniquely ensuring conservation of mass.



(a) Node based (b) Cell based

Figure 7.1: Control volumes at a non-uniform block transition

Similar advantages can be seen for combined free-surface- and FSI-simulations as shown in Fig. 7.2. In the node-based layout (Fig. 7.2a) the middle node is a solid (and inactive) node in the LB context and does not provide valid particle distribution functions, which are needed for the advectional step. A cell-based layout cures this problem (Fig. 7.2b). The solid fraction of the boundary cells has to be determined, but the basic algorithm is applicable.



(a) Node based (b) Cell based

Figure 7.2: Control volumes next to a rigid body

The new layout leads to additional requirements for the consistency between cell and node states. A cell is fluid if and only if all its vertices are fluid nodes. A cell is gas if and only if all its vertices are gas nodes. All other cells are interface cells.

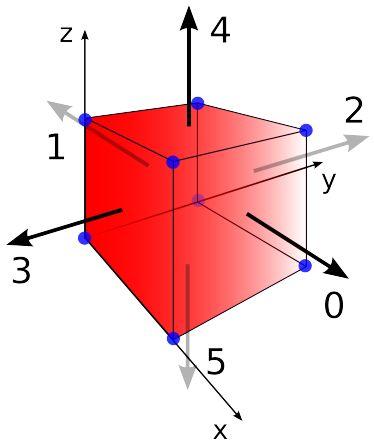


Figure 7.3: VOF control volume with eight LBm grid nodes and six main exchange directions

7.1.1 Extensions to the LBM

The LB solver has to be modified to incorporate the free surface capturing capabilites. From the numerical point of view, a free surface represents a moving boundary. The transient computational domain changes, and inactive lattice nodes outside the fluid domain exist. Hence, an additional flag field serves to observe if a node is inside or outside the fluid domain and, resp., has to execute collision and propagation steps. At the free surface, a pressure boundary condition is applied, in analogy to section 3.4.4. Apart from the prescriped boundary value for the atmospheric pressure, the free surface velocity for the anti-bounce back along the lattice link i is obtained from the fluid velocities inside the domain by linear extrapolation:

$$\mathbf{u}_{bdy,i} = \mathbf{u}(\mathbf{x}) + 0.5\,(\mathbf{u}(\mathbf{x}) - \mathbf{u}(\mathbf{x} - \mathbf{e}_i \Delta t)) \tag{7.1}$$

The correct velocity extrapolation is crucial to the final free surface results, especially in regions with high gradients and high interface curvature as e.g. breaking waves.

A very common challenge of Eulerian methods with time-dependant computational domains is the activation of new fluid nodes, as the numerical front evolves. These nodes change their state from inactive (gas) to active (fluid). Hence, no valid simulation data (i.e. distribution functions in the LBM) is available at the concerning nodes. In order to obtain a smooth solution of the flow and to avoid unphysical shock waves in pressure, velocity or higher order moments, a meaningful initialization of new fluid nodes has to be done. As a first guess for new nodes, the macroscopic properties of the surrounding nodes are interpolated:

$$\bar{\rho} = \frac{\sum_i w_i \cdot \rho_i}{\sum_i w_i} \quad \text{and} \quad \bar{\mathbf{u}} = \frac{\sum_i w_i \cdot \mathbf{u}_i}{\sum_i w_i} \tag{7.2}$$

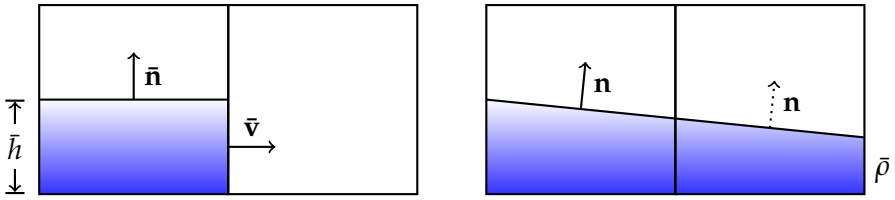
with a weighting factor w_i, typically set to 1.0. Then, the equilibrium distribution functions are attached to the new node (section 3.8). Subsequently, a local Poisson-type iteration serves to improve the local pressure and the non-equilibrium parts of the distributions functions ([114], Eq. 3.35).

7.1.2 Extensions for the advection scheme

Similar extensions have to be done for the advection scheme, especially concerning the time stepping, the treatment of interface cells next to solid boundaries and the boundary conditions, which have not been considered in the previous chapter yet.

Boundary conditions

At the free surface boundary, the kinematic boundary condition is inherently fulfilled by the advection scheme. The interface is advected in a Lagrangian way, so that the total flux through the interface surface is zero. At solid boundaries, a no-flow boundary condition guarantees that no fluid leaves the cell: if a face of the control volume contains solid nodes only, the resulting flux is set to zero. At boundaries where a flux q is prescribed, the information is split up. The velocity v is used for the velocity boundary condition of the flow solver, and the corresponding water height h is used as boundary condition for the advection scheme: the water level $\bar{h}$ and the interface normal $\bar{\mathbf{n}}$ in the boundary cells are not allowed to change. They are fixed to the desired boundary value (Fig. 7.4a). At pressure boundaries of the fluid solver, which are mainly used as outflow boundary conditions, zero-gradient conditions for the surface normal are assumed. Alternatively, if the flow direction is clear, a backward estimation of the surface normal can be used (Fig. 7.4b).



(a) Inflow (b) Outflow

Figure 7.4: Inflow and outflow boundary condition

Interface cells at the boundary

For interface cells in the vicinity of solid walls, several model extensions have to be introduced in addition to the no-flow boundary condition in wall normal direction. At first, the size of the control volume is changed, as a cell potentially contains parts of the solid phase, in addition to gas and fluid. For this purpose, the *solid fill level* $\zeta(\mathbf{x})$ is introduced and qualifies the relative amount of the cell volume that is occupied by solid obstacles. Consequently, Eq. 6.11 has to be modified according to

$$\varepsilon^{t+1} = \frac{\varepsilon^t \rho^t \zeta - \sum_i \Phi_i \cdot \Delta t}{\rho^{t+1} \zeta}. \tag{7.3}$$

Additionally, if an edge of the control volume is cut by a solid obstacle, the mean solid volume in the corresponding cell is evaluated and serves as a weighting factor for the flux term Φ_i:

$$\Phi_i^{T \to T} = \Phi_i \cdot \left[1 - \frac{\zeta(\mathbf{x}) + \zeta(\mathbf{x} + \mathbf{e}_i \Delta t)}{2} \right] \tag{7.4}$$

This approach is comparable to the simple surface reconstruction scheme for the fluid surface, as given in Eq. 6.13. Finally, the finite difference evaluation of the surface normal fails for boundary cells. The application of centered finite differences is not possible, since the solid cell does not contain valid information on the fill level, so that asymmetric finite differences are used.

Adaptive time stepping

The resulting maximum velocity magnitude in a LB simulation typically is limited by the Mach number, which controls the convergence of the LB scheme to the incompressible Navier-Stokes equations. Reasonable Mach number values are in the range of 0.005 to 0.05, corresponding to maximum LB velocity magnitudes from ≈ 0.0025 to 0.025. An adaptive time stepping for the advection scheme has been introduced in order to save computational costs, since the advection scheme has shown to produce good results even at CFL numbers of around 0.1 or higher (see section 6.3.3). At runtime, the maximum global velocity magnitude is evaluated in certain intervals. Then, the time interval for the interface update is changed and the velocities are rescaled before the evaluation of the flux terms according to

$$t_{UI} = \frac{\mathrm{CFL}_{max}}{\mathrm{CFL}_{actual}} \quad \text{and} \quad \mathbf{v}_{UI} = \frac{t_{UI}}{t_{LB}}\mathbf{v}_{LB} \tag{7.5}$$

For typical LB simulations, the interface can be updated each 35 to 350 time steps only, without substantial negative impact on the overall result quality. Especially for gravity-driven simulations, as e.g. falling drops or dambreak scenarios, where the fluid is accelerated from a state of rest, the performance gain during the initial stages of the simulation is high.

7.1.3 Calculation of cell densities and face velocities

The advection scheme relies on cell-centered values of fill level ε and mass m of one cell. Hence, an averaged cell density ρ is needed to convert between these intensive and extensive quantities. It is obtained by averaging the fluid densities of all cell fluid nodes according to

$$\rho^t_{cell} = \frac{1}{n_f}\sum_{n_f}\rho^t_i, \tag{7.6}$$

for a cell with n_f fluid nodes. A more sophisticated interpolation using information on the exact fluid distribution in a cell (by means of a *node fill level*) did not further improve the results.

The treatment and calculation of free surface velocities has to be done with great care. In common VOF methods, the advection velocities are defined on the face centroids of the control volume. In most of the cases, the same staggered grid layout is also used for the solution of Navier-Stokes solutions, so that inter- or extrapolation is not needed. Opposite, in the hybrid LB-VOF method, the fluid velocities are known on the eight lattice nodes in the corners of a cell. If the cell is not entirely filled with fluid, gas nodes appear, which do not carry valid particle distribution functions and hence no valid velocity information. In a low-order ansatz, the face values of density ρ_i and velocity v_i are determined from the corresponding nodal values:

$$\mathbf{v}_i = \sum_i \mathbf{v}_i \cdot \hat{\mathbf{n}} \quad \forall \text{ fluid nodes } i \tag{7.7}$$

In the worst case of an underresolved wave front, this interpolation is constant in space, and grid refinement will be necessary to improve accuracy. For all other cases, the interpolation is of linear order, which is sufficient since it corresponds to the order of the surface reconstruction scheme, discretization in time, and the pressure boundary condition. For testing purposes, an extrapolation approach which is consistent with the calculation of the free surface velocity for the pressure boundary condition has been implemented: if a face is a fluid face and enough fluid nodes are available, linear interpolation serves to obtain the corresponding velocity value. Again, the results were not significantly improved.

7.2 Treatment of entrapped air

The main drawback of free surface models in comparison to multiphase models is the neglection of the second (air) phase. The air is represented by a boundary condition for the surrounding atmospheric pressure. Hence, as soon as parts of the air phase are completely surrounded by the water phase, the physics are not modeled correctly anymore. The fluid pressure is usually higher than the surrounding atmospheric pressure due to the hydrostatic pressure contribution. The bubbles would be compressed and would finally disappear, as their internal resistance against compression is not modeled. To cure this, at least some of the air characteristics might be included in the simulation. In a very primitive approach, the basic behaviour of the air phase is imitated, using the relations for an isothermal change of state of an ideal gas:

$$p\ v = \text{const.} \tag{7.8}$$

Consequently, a compression of air leads to an increase in pressure and the free surface boundary condition in the entrapped air regions is updated. This approach does not provide a detailed pressure distribution along the phase interface, but the handling of entrapped air is improved. From the algorithmic point of view, the closed, entrapped gas regions have to be detected and the change of gas volume is udpated in a certain update interval. Donath et al. [30] propose an improved version for the localization of enclosed gas regions and its implementation in an LBM context. Initial validation is shown in section 7.5.1.

7.3 Resulting update interface algorithm

After the LB collision and propagation, the position of the phase interface is updated on the basis of the new, valid velocity and density fields. The whole algorithm loops over the *interface* cells only, as is characteristic for geometry-based advection schemes. The first part of the update interface algorithm evaluates the mass fluxes for all interface cells. If an interface cell changes its state (i.e. runs full or empties), it is added to a list of new gas or new fluid cells, together with the excessive mass. In the second part of the interface update, these changes are applied to the state matrix and consistency between nodal state and cell fill level is assured. In the loop over the new cells, the eight nodes of the cell are checked and - in case they change their state - they are stored in a list of new gas or fluid nodes. The number of new interface neighbors is counted to determine and store the portion of mass for each new interface cell. Afterwards, the new fluid nodes have to be initialized, as they

do not contain any distribution functions. After interpolating the macroscopic values of density and velocity from neighboring old fluid nodes, the non-equilibrium part of the distribution functions is improved with a local Poisson-type iteration [114]. The whole algorithm is of Jacobi-type. The result does not depend on the order of loop execution, as a temporary fill level matrix is used to store the values of ε_i^{t+1}.

7.3.1 Grid refinement

Compared to the pure advection test cases in the previous chapter, grid refinement for the full free surface kernel is more demanding. Apart from space refinement, the nested LB time stepping scheme leads to new challenges at the non-uniform grid interface. So far, a simplified adaptive grid refinement technique is used, in which the phase interface resides on the finest grid level. As the surface reconstruction scheme is of the order $\mathcal{O}(\Delta x)$ resp. $\mathcal{O}(\Delta t)$, and the dynamic free surface boundary condition also is of the order $\mathcal{O}(\Delta x)$, this limitation seems to be reasonable. During the simulation, a phase interface approaching the borders of a refinement region is detected, and neighboring regions are refined before the phase interface enters that region.

Nevertheless, the non-uniform grid refinement of the advection scheme has to be demonstrated. In the long run, in combination with higher-order pressure boundary conditions and higher-order surface reconstruction schemes, the extension of the grid refinement to phase interfaces across different refinement levels is crucial.

7.4 Validation

For validation purposes, several test cases have been examined. Apart from the comparison to test case-specific reference data, the conservation properties of the scheme have been checked continuously. The advection scheme itself has proven to be mass-conserving. In the coupling to a flow solver, the additional interpolation rules for the face velocities, cell densities and the initialization of new fluid nodes and the distribution of mass might possibly affect the conservation properties of the overall method. Hence, the conservation of mass is observed for all fluid and interface cells via

$$\mathcal{M} = \int_\Omega \varepsilon\rho = \sum_i \varepsilon_i \rho_i \tag{7.9}$$

At first, two- and three-dimensional breaking dam benchmarks show that the hybrid model is capable of simulating real world civil engineering fluid applications. Solid objects in the flow demonstrate that the basic approach to account for solid bodies works fine. Wave impact on these solid boundaries is checked and validated, discussing the treatment of corner points and geometrical discontinuities. The breaking dam test cases do not require the use of elaborate boundary conditions, so that after initial breaking dam validation, the behaviour of a free falling water jet is analyzed to show the usage of proper inflow and outflow boundary conditions. After that, several show cases on different time- and length scales will be presented to underline the wide applicability of the solver.

Algorithm 7.1: Update interface algorithm (PLIC surface reconstruction)

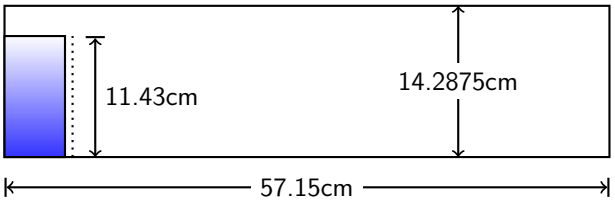
```
if node type == fluid then
    collision;
    forcing;
    propagation;
    apply boundary conditions;
end
if cell type == interface then
    determine face values for density ρ_i and velocity u_i;
    determine surface normal vector n;
    reconstruct surface (i.e. determine plane parameter α);
    determine face fill levels ε_i;
    calculate mass fluxes Φ;
    evaluate new fill level ε_i^{t+1};
    if new fill level ε_i^{t+1} < 0.0 then
        add cell to container with new gas cells;
        set new fill level to 0.0 and store excessive mass portion;
    end
    if new fill level ε_i^{t+1} > 1.0 then
        add cell to container with new fluid cell;
        set new fill level to 1.0 and store excessive mass portion;
    end
    set new cell states;
    check consistency of node and cell states;
    set new node states;
    initialize new fluid nodes;
    check and adjust interface cell states;
    if applicable, locally distribute the excessive mass (Eq. 6.26);
end
```

7.4.1 Breaking dam

Similar to the GPU code, the classic breaking dam benchmark [112] is used for the initial validation. The main setup and the simulation parameters are shown in test case setup 7.1. On the front, bottom and back wall slip boundary conditions are used, while for the left and right wall, periodic boundary conditions are applied. For the LB an incompressible MRT model with LES was chosen (section 3.7, [98]). The calculations are stopped as soon as the surge front reaches the back wall of the container.

Param.	Value
Re	103483
Fr	2.418
U_{max}	1.71 [-]
Domain	0.5715m x 0.142875m
	(22.5in x 5.625in)
Water column	2.25in x 4.5in
Lattice	128x1x32
	256x1x64
	384x1x96
	512x1x128
	1024x1x256

(a) Parameters

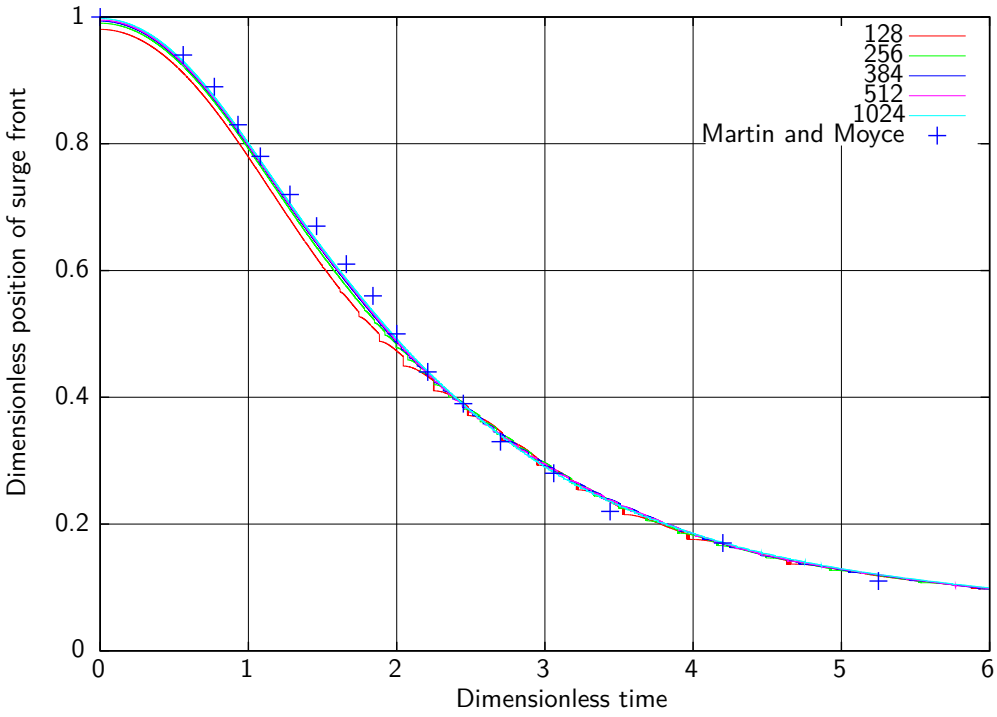


(b) Geometry

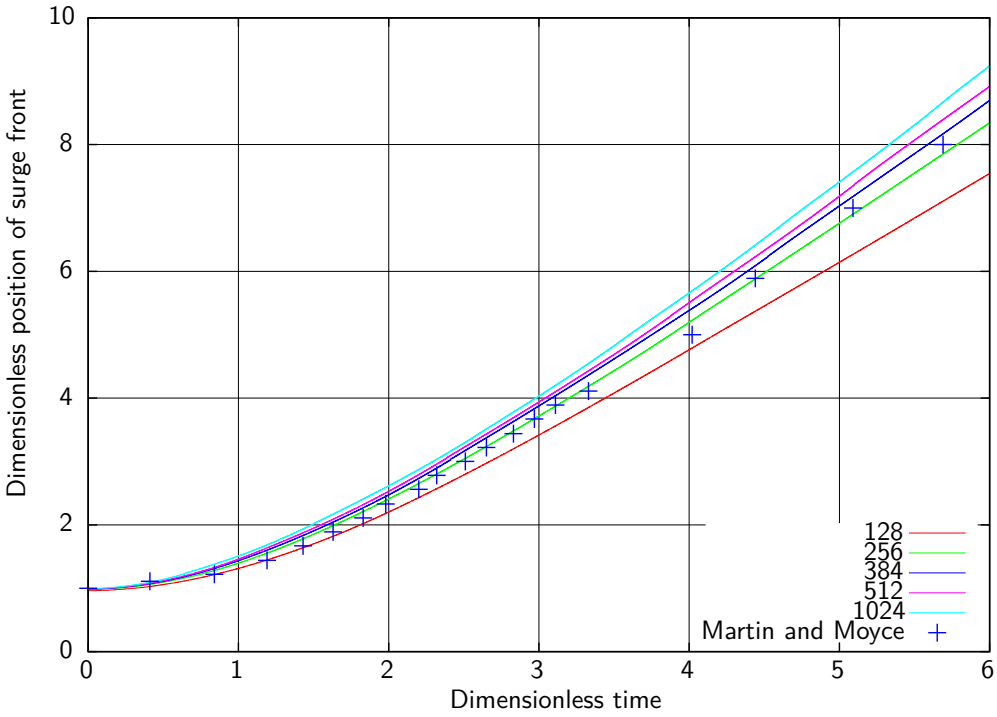
Test case 7.1: Breaking dam (reloaded)

This test case is demanding in terms of boundary conditions at the triple point, where both the free surface boundary condition and the no-slip or slip constraint from the bottom wall has to be fulfilled. If the boundary conditions are imposed improperly, the resulting fluid velocity at the corresponding node is too low. The resulting interface deformation is incorrect as the penultimate layer of interface cells is filled up before the bottom one. Consequently, the surge front would evolve in a staircase-shaped way. To avoid this, a consistent macroscopic velocity information is needed at the triple point. The same velocity value should be used in the evaluation of both slip and free surface boundary condition. This problem did not show up in the GPU code, since the slip boundary condition was realized with an implicit bounce forward scheme.

In Fig. 7.5a and Fig. 7.5b, the results for four different grid resolutions are compared to the experimental reference data from Martin and Moyce. Very well agreement concerning the height of the collapsing water column can be observed, also in the comparison to the results of the GPU implementation in Fig. 5.8a, whereas the numerical surge front again evolves slightly faster than the experimental one. This observation is consistent with the results of various other groups, e.g. Sauer [141], Kölke [99], Salih and Moulic [139], and our previous GPU simulations in section 5.4.1.



(a) Position of top of the water column



(b) Position of the surge front

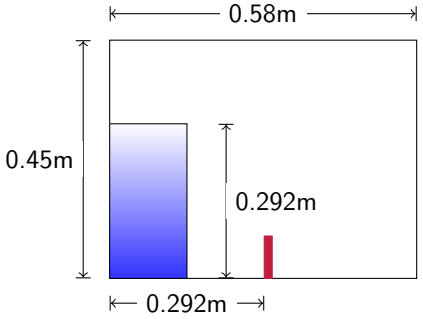
Figure 7.5: Results for dambreak test case

7.4.2 Breaking dam with obstacle

After the initial validation, the classic breaking dam benchmark is extended and a a solid obstacle is added in the center of the domain. The collapsing of the water column starts, the front evolves and finally impacts on a solid wall. As soon as the solid obstacle is reached, the flow pattern changes drastically, comparable to wave impact incidents. After the water descends, a small, highly curved jet evolves, and the water is deflected to the top. Corresponding experiments have been carried out by Kölke [99]

Param.	Value
Domain	0.58m x 0.45m
Water	0.146m x 0.292m
Obstacle	0.015m x 0.08m
Lattice	58x1x45
	116x1x90
	232x1x180
	464x1x360
Δx	0.01m, ..., 0.00125m

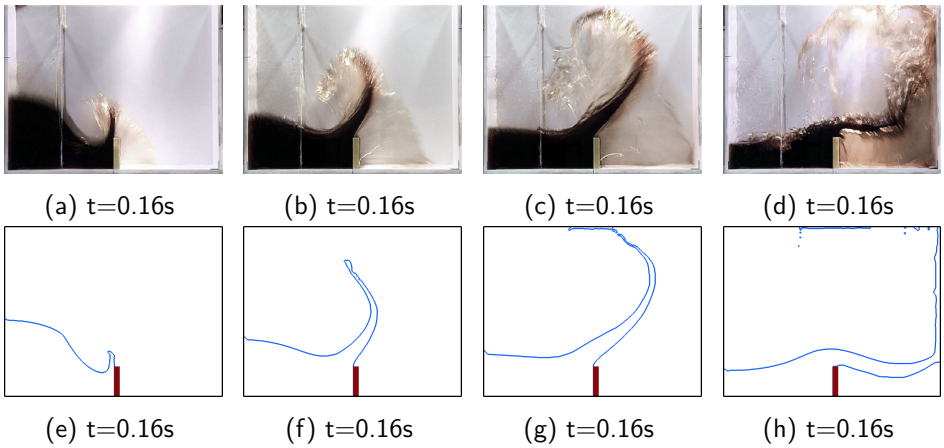
(a) Parameters



(b) Geometry

Test case 7.2: Breaking dam with solid obstacle

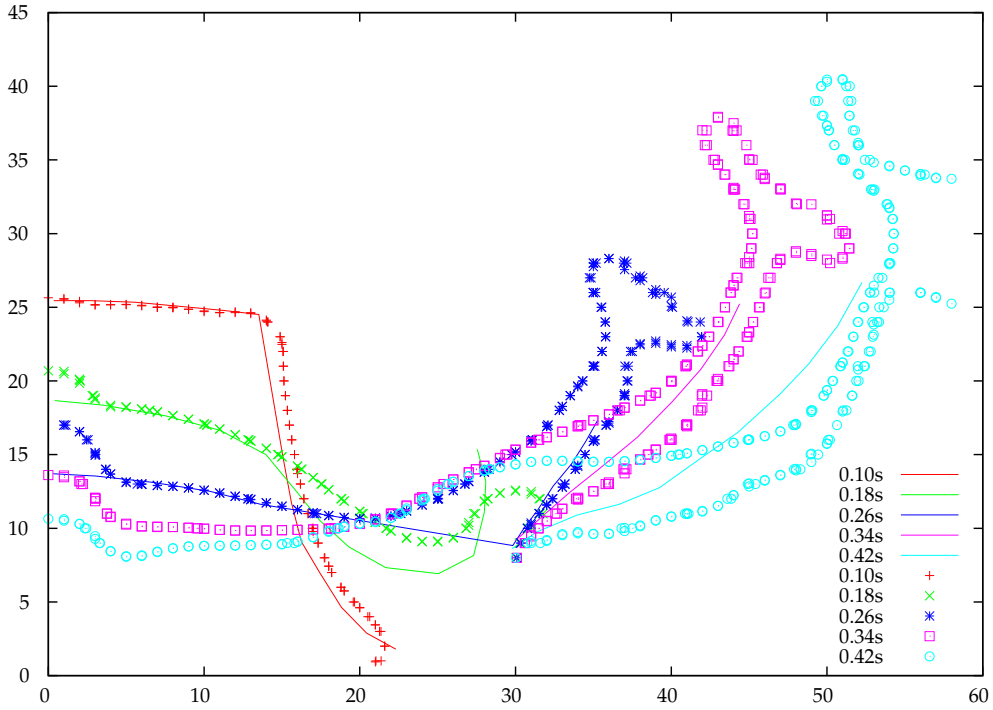
A set of snapshots is given in Fig. 7.6 for four selected points in time. In Fig. 7.7, the numerical results are compared to experimental observations by [99]. Very good agreement can be seen, at least for this qualitative comparison. The jet evolves slightly faster than the one in experiment, but the time-lag decreases with grid refinement. For the LB, a careful treatment of the slip-slip corner boundaries is crucial for acceptable results. For slip BCs the normal vector information on the LB nodes is needed, which is discontinuous in corner points. If a smoothed version is used, where the corner point uses a linear interpolation of the neighboring surface normal vectors, the corner is "rounded out" and the jet shape is falsified. For this reason, the normal vector at the box corner is set to the normal vector of one of the participating faces. A proper jet shape is obtained.



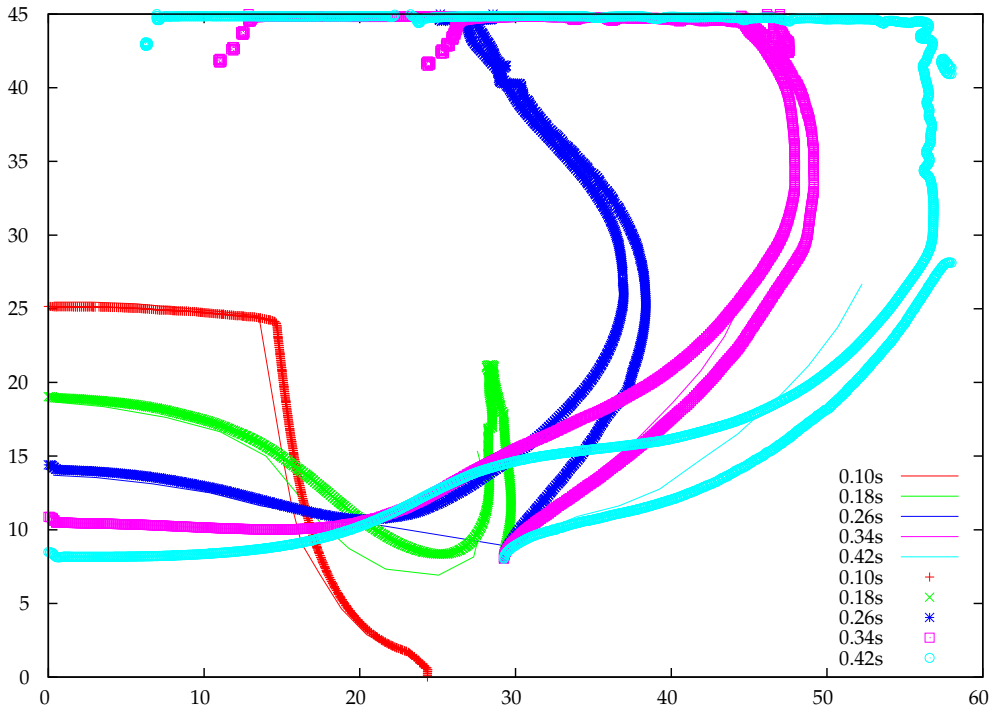
(a) t=0.16s (b) t=0.16s (c) t=0.16s (d) t=0.16s

(e) t=0.16s (f) t=0.16s (g) t=0.16s (h) t=0.16s

Figure 7.6: Experimental results of [99] compared to numerical results



(a) Coarse grid (58x1x45)



(b) Fine grid (464x1x360)

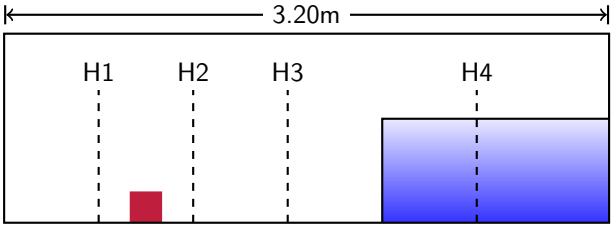
Figure 7.7: Results for dambreak test case with obstacle for two different grid resolutions

7.4.3 Breaking dam with obstacle and pressure probes

The following test case serves to quantify the free surface-structure interaction results in terms of water height and impact pressure, after the previous qualitative validation of jet positions. The corresponding experiment has been conducted at the Maritime Research Institute Netherlands (MARIN) [94, 173]. A scale tank with an open roof is used (test case setup 7.3). In analogy to the two-dimensional experiments, a water column in the back part of the tank (behind x = 2 m) is going to collapse. Opposite to the thin diaphraghm of Martin and Moyce [112], the water is column is constraint by a door which is pulled up by releasing a weight. The water heights in the tank are observed at four positions (H1 in the reservoir and H2, H3, H4 in the tank at $x_i = 0.5, 1.0, 1.5, 2.66m$).

Param.	Value
Domain	3.20m x 1.00m
Water	1.20m x 0.55m
Obstacle	0.16m x 0.16m
Lattice	256x1x80 ($\Delta x = 0.0125m$) 512x1x160 ($\Delta x = 0.00625m$)

(a) Parameters

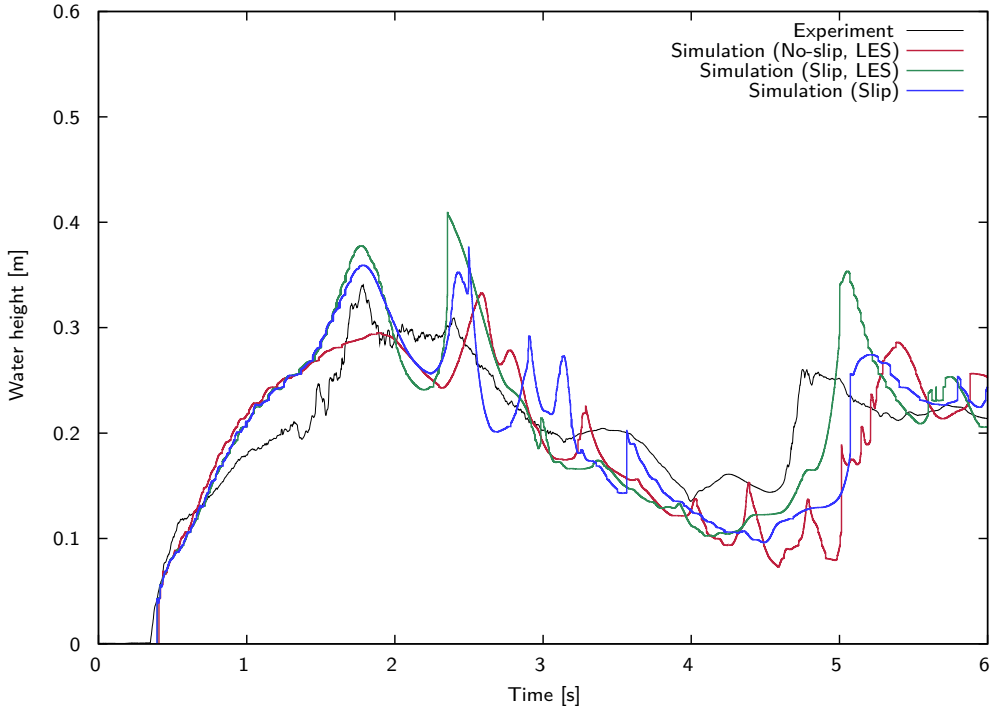


(b) Geometry

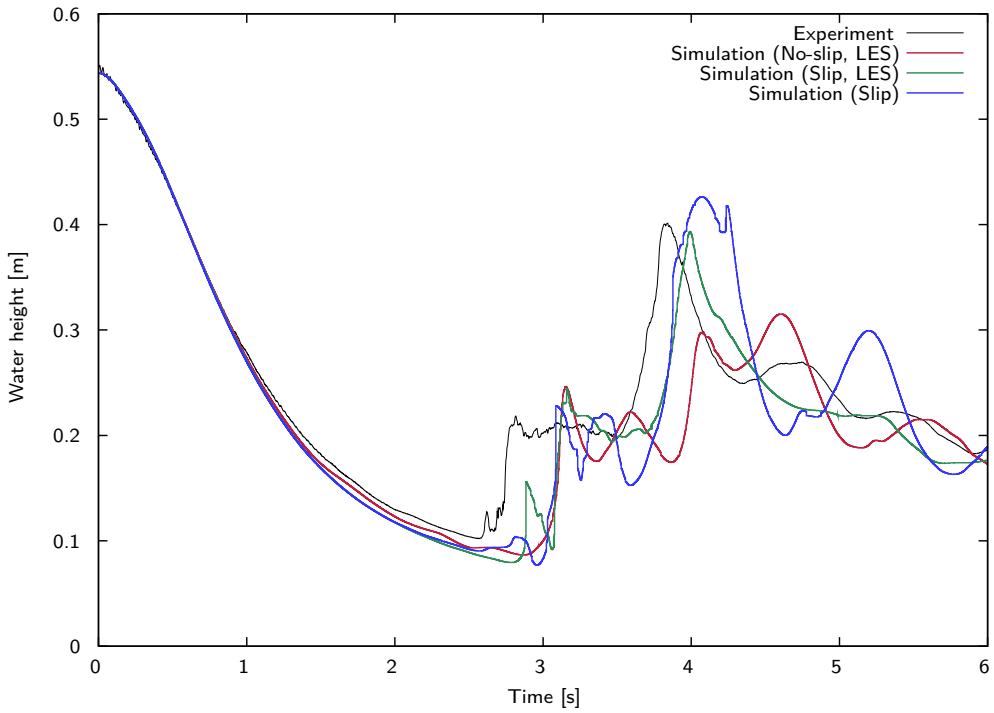
Test case 7.3: Breaking dam (3D)

The tank is used for at least two sets of experiments. In the first one, performed by [94], a box has been placed in the tank and was covered by eight pressure sensors, four on the front of the box at height z = 0.025, 0. 063, 0.099 and 0.136, and four on the top of the box at x = 0.806, 0.769, 0. 733 and 0.696. The experimental results for the height probes H2 and H4 and the pressure probes P1 and P7 are public domain and can be downloaded [166]. The same experimental setup, without the box, is used by Wemmenhove [173] to examine tank sloshing phenomena with flexible walls.

A 2D equivalent test case is used for the numerical simulation, assuming periodicity in the third space direction. In Fig. 7.8, the results for the height probes H2 and H4 on a grid with 256x1x80 nodes are compared to the reference data for three different configurations. In general, good agreement can be observed. To be more precise, the initial peak at height probe H2 can not be reproduced if no-slip bottom boundary conditions are assumed, whereas the simulation with frictionless bottom overpredicts the impact height. The arrival time of the reflected wave at $t \approx 5s$ differs about $0.4s$ in all three simulations. It is noteable that the simulation without bottom friction but with LES model predicts a higher value for the reflected wave amplitude, compared to the slip-simulation without LES. The results for the height probe H4 show similar characteristics, including a comparable time-lag. The pressure probes P1 and P7 are evaluated for a no-slip LES simulation on the fine grid (512x1x160 lattice nodes), and the results are depicted in Fig. 7.9. The pressure at probe P1 qualitatively corresponds to the experimental data and the delay of initial and reflected wave is small. The amplitudes match reasonably, although the pressure signals of the simulation were processed with a low-pass filter to remove the pressure noise. A filter width of $0.03s$ has been chosen. At pressure probe P4, a similar flow behavior is reproduced too. For future work, three-dimensional simulations of this test case are recommended in order to improve the prediction of the reflected wave signal.

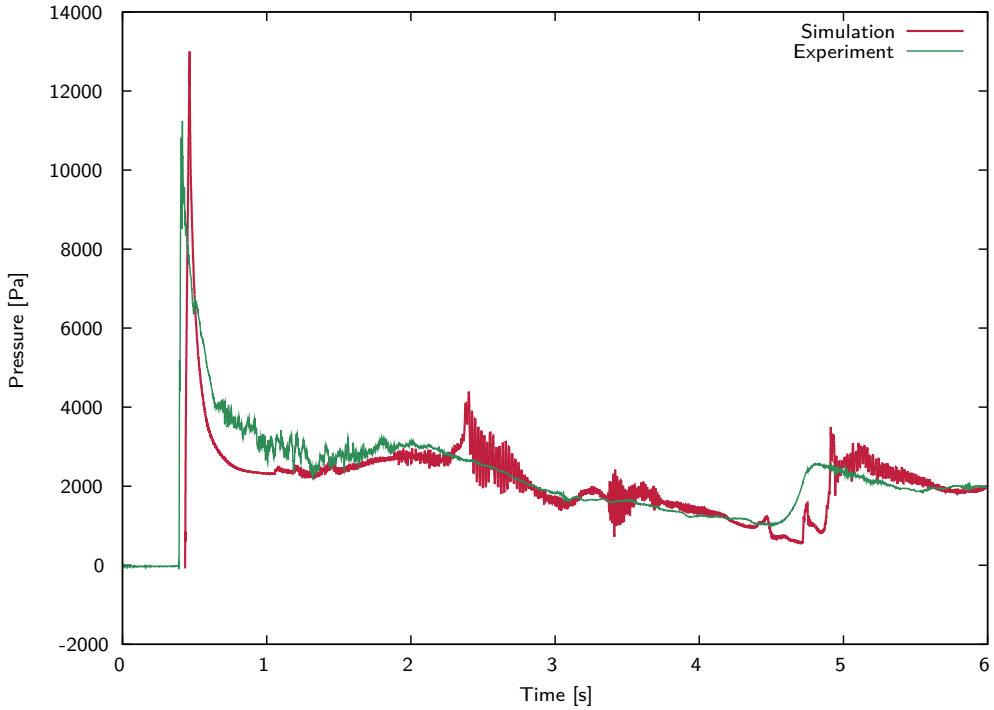


(a) Water level probe H2

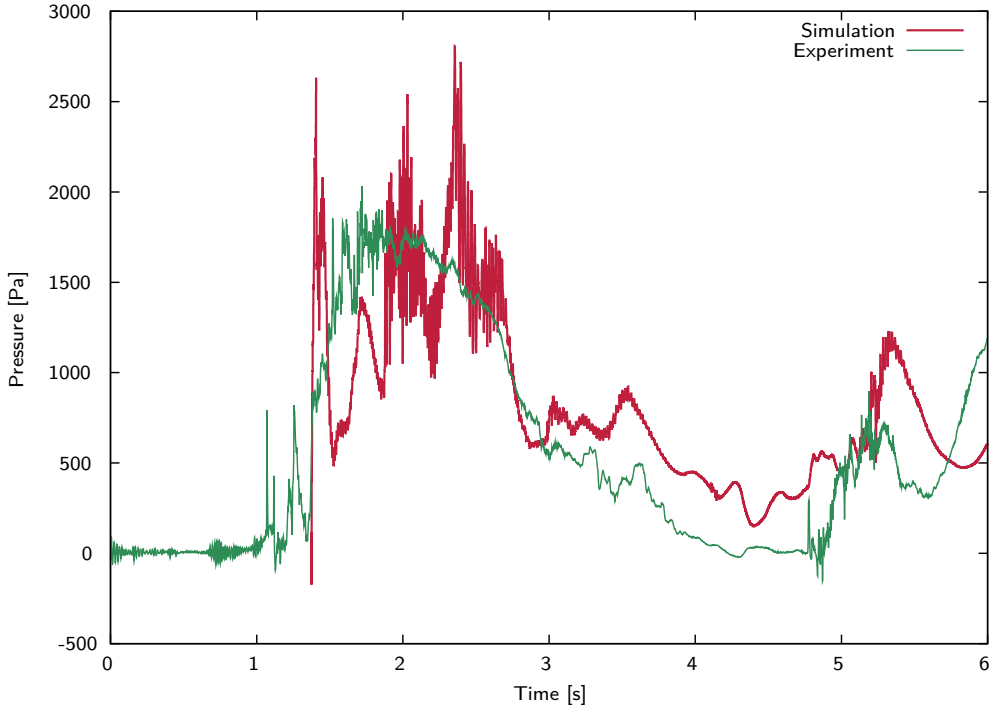


(b) Water level probe H4

Figure 7.8: Results for height probes H2 and H4 for three simulation setups ($\Delta x = 0.0125m$)



(a) Pressure probe P1 (low-pass filtered)



(b) Pressure probe P7 (low-pass filtered)

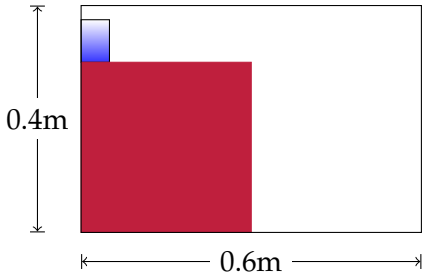
Figure 7.9: Results for pressure probes, high-resolution simulation with LES and no-slip boundary conditions ($\Delta x = 0.00625m$)

7.4.4 Free jet

Test cases on the basis of breaking dam scenarios are demanding in terms of free surface advection, pressure evaluation and representation of obstacles in the flow. In contrast, complex boundary conditions are not needed as the entire domain is surrounded by impermeable no-slip or slip-walls. Hence, for testing the inlet and outlet boundary conditions, a free jet over a horizontal slip-surface is examined. Initially, the domain is empty and an inflow on the left is specified. The inflow is set to $q = 0.0643m^2/s$ per unit width. The inflow water depth $h = \kappa y_c$ is set depending on the critical water depth $y_c = 0.075$. We examine the cases $\kappa = 0.49, 0.68, 1.00$, corresponding to the Froude numbers Fr $= 1, 1.78$ and 2.91. Gravity is given as $g = 9.81m/s^2$. Note that a full three-dimensional movement of the free surface is possible and the simulations starts with an empty domain.

Param.	Value
Domain	0.6m x 0.4m
κ	0.49,0.68,1.00
Fr	1, 1.78, 2.91
Lattice	120x1x80
	240x1x160
	480x1x320
Δx	0.5cm, 0.25cm, 0.125cm

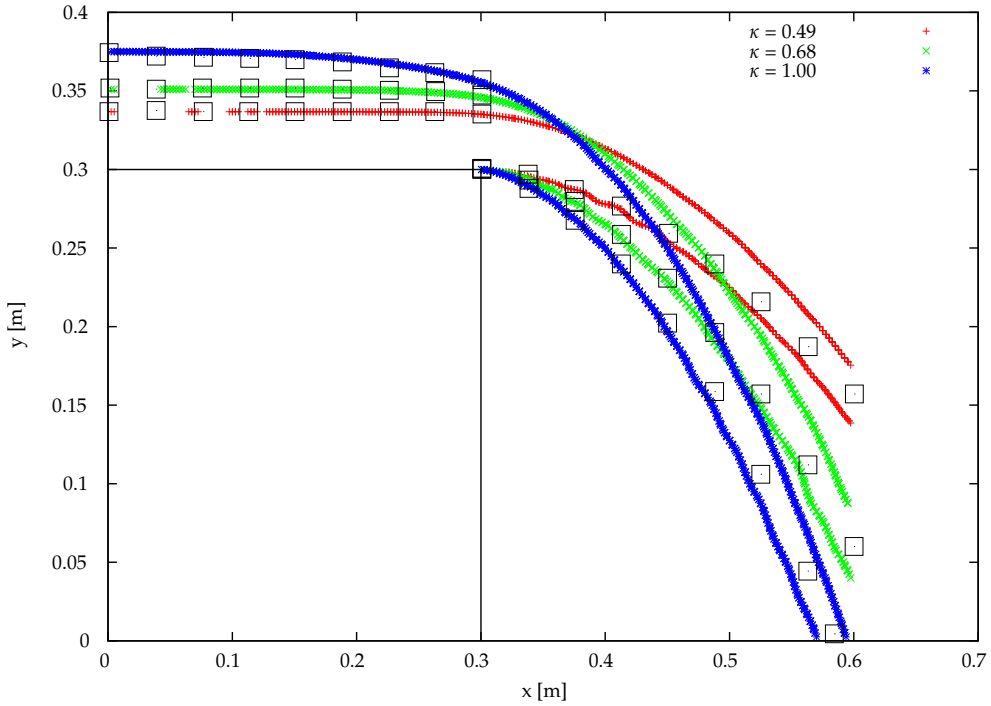
(a) Parameters



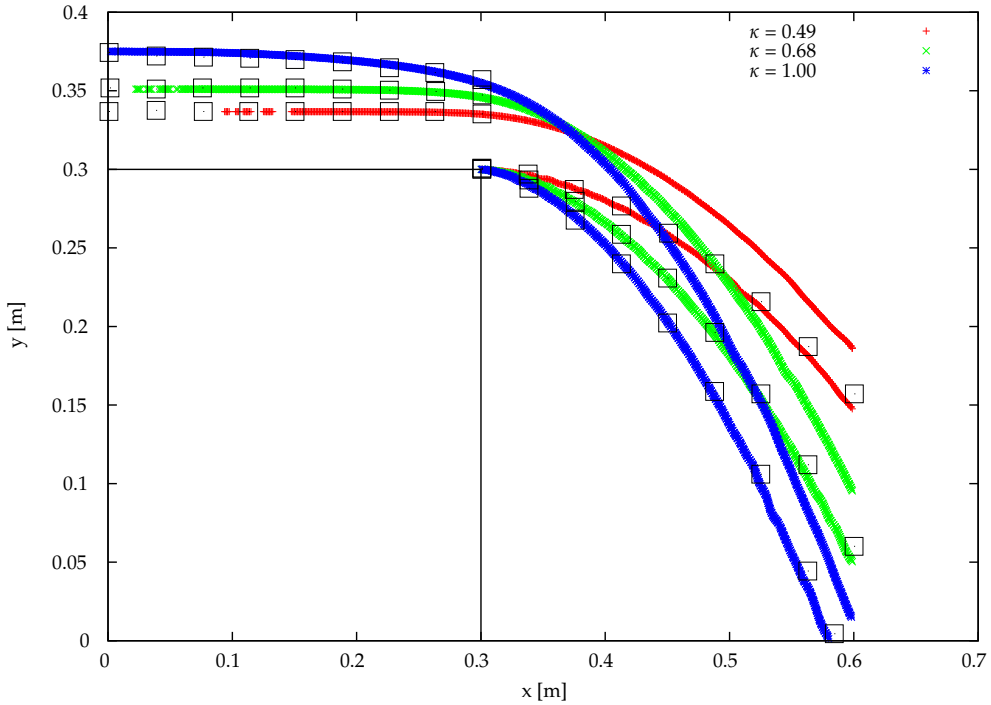
(b) Geometry

Test case 7.4: Free jet

Water flows into the domain. After the jet passes the solid obstacle at $x = 0.3m$, it falls freely until a quasi-static position is reached. During the free fall, the fluid is accelerated in vertical direction and the jet is narrowed. In Fig. 7.10, the resulting free surface displacements in comparison to reference data are given. Very well agreement can be observed for all three cases. Moreover, in comparing the results for two different grid resolutions, the convergence to the experimental data is evident. The free jet position on a coarse grid (Fig. 7.10a) is lower than the experimental values, indicating that the horizontal velocity component is too low. Further grid refinement leads to a higher jet position. Again, aproper handling of the surface discontinuity at $\mathbf{x}_s = (0.3, 0.3)$ is crucial in this test case. If the corner was rounded, the jet would stick to the wall and the total jet shape would be distorted. For this reason, the normal vector at the singularity $\mathbf{x}_s$ is assumed to be $\mathbf{n}_s = (0, 1)$.



(a) Coarse grid



(b) Fine grid

Figure 7.10: Position of the free jet for two different grid resolutions in comparison to reference data (black squares, [110])

7.5 Show cases

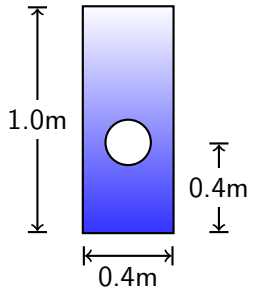
After this basic validation, the free surface algorithm is applied to several applications in the field of civil and structural engineering. The simulation of weirs with hydraulic jump, sloshing tanks and wave impact on naval structures are possible. Opposite to the GPU algorithm which was presented in chapter 5, the VirtualFluids implementation is especially suitable for the handling of complex geometries, non-uniform grids and fluid-structure interaction. Hence, in the following two chapters (chapter 8 and chapter 9), more sophisticated applications are presented, whereas here only show cases are shown.

7.5.1 Rising bubble

Free surface flow simulations without a simulation of the air-phase are in the center of interest of this work. Nonetheless, the presented VOF approach with PLIC interface reconstruction can be applied to multiphase problems too. A rising bubble is examined to show the general suitability of the approach to cope with two-phase simulations. Surface tension is neglected, and the air phase is not simulated, but represented as a closed volume obeying the ideal gas law according to section 7.2.

Param.	Value
Domain	0.4x0.4x1.0
Sphere	$\mathcal{C} = (0.2, 0.2, 0.4)$, $r = 0.1$
Lattice	20x20x50
ν_{LB}	0.01
g_{LB}	-0.00002

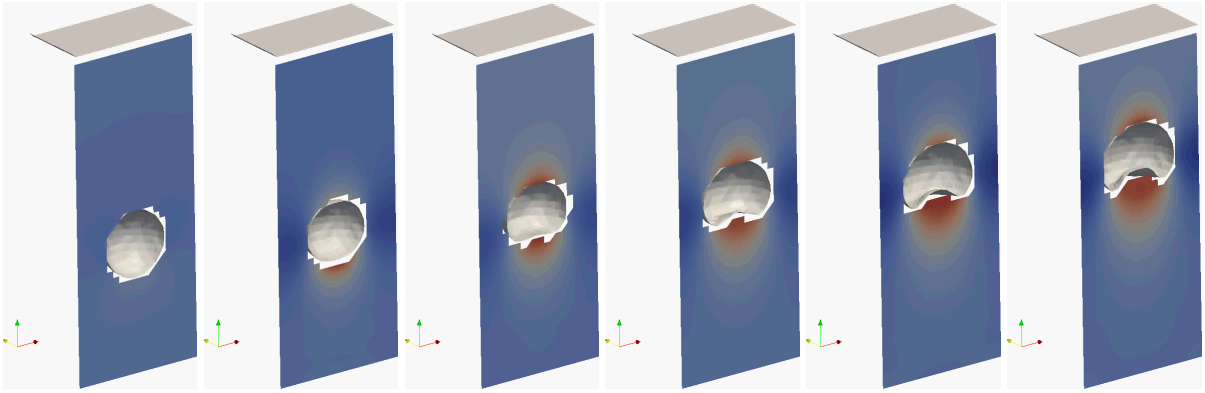
(a) Parameters



(b) Geometry (2D cut)

Test case 7.5: Rising bubble (without surface tension)

Detailed comparisons to reference data for bubble rising are meaningless, as no surface tension is present. Nonetheless it can clearly be seen that a smooth interface is obtained and the general expected flow behaviour is well represented. The bubble starts to rise and expands at the same time, due to the lowering hydrostatic pressure.



(a) $t' = 0.0$ (b) $t' = 0.7$ (c) $t' = 1.4$ (d) $t' = 2.1$ (e) $t' = 2.8$ (f) $t' = 3.5$

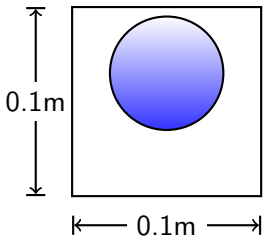
Figure 7.11: Rising sphere, selected time steps (dimensionless time $t' = t\sqrt{g/h}$)

7.5.2 Falling drop

Falling drops are nice show cases due to the varying flow states during the falling process. Before the impact, the fluid usually is in a subcritial state with a Froude number lower than one. After impact, high-velocity regions appear, which even may lead to surface break up or small droplets, leaving the closed water surface.

Param.	Value
ν	$1 \cdot 10^{-6}$
g	9.81
Domain	0.1x0.1x0.1
Sphere	$\mathcal{C} = (0.05, 0.05, 0.065)$
	$r = 0.03$
Lattice	64x64x64

(a) Parameters

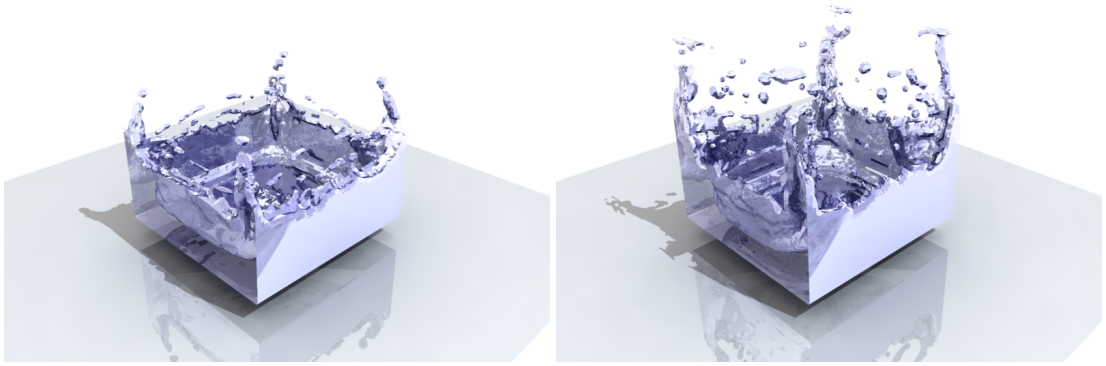


(b) Geometry

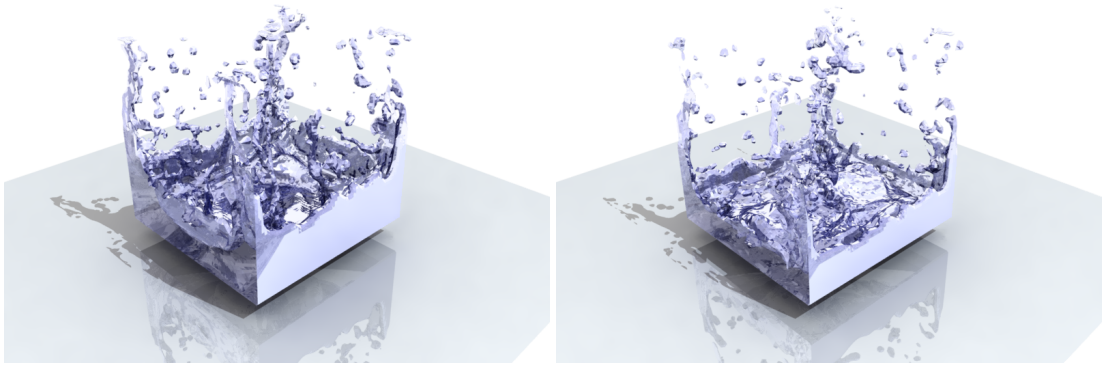
Test case 7.6: Falling drop

In Fig. 7.12, the results for six selected time steps are given. For the visualization, the 3D graphic software Blender is used. During the numerical simulation, an isosurface for the cell fill level $\varepsilon = 0.5$ is generated via a classical marching cubes algorithm. The available information on the PLIC interface in terms of cell normal vectors and the plane constant α are not used. The resulting surface mesh would be discontinous at the transition between two cells and, consequently, useless for raytracing purposes in the postprocessing step. The surface triangle mesh is transferred to Blender by means of the X3D file format. The Blender Python interface serves to automatically read and render whole sequences of simulation data, once the general scene setup with geometry, material properties for rendering and light scenes is done.

(a) $t' = 0.8$ (b) $t' = 1.6$



(c) $t' = 2.4$ (d) $t' = 3.2$



(e) $t' = 4.0$ (f) $t' = 4.8$

Figure 7.12: Falling drop, selected time steps (dimensionless time $t' = t\sqrt{g/h}$)

7.5.3 Breaking dam using adaptive grid refinement

Exemplarily, the breaking dam benchmark is examined using adaptive grid refinement around the phase interface. The LB solver VirtualFluids (see chapter 10) is based on block-structured grids. The computational domain is partitioned into a set of blocks, each of which contains a specific number of lattice nodes at a constant grid spacing Δx. At the block interface, information is exchanged. If neighboring blocks have different grid resolutions, the scaling and interpolation techniques of section 3.9 are used. The grid refinement is implemented on the basis of a block-wise strategy: blocks are selected for the grid refinement at runtime, and each of the coarse blocks is replaced by eight fine ones. In the grid coarsening step, in return, each group of eight fine blocks that are selected for the coarsening is replaced by one coarse block, see also Freudiger [42]. For the free surface implementation, the phase interface is required to reside on the finest grid level, i.e. in the blocks with highest grid resolution. As the numerical front evolves, blocks will be refined and coarsened dynamically.

The performance gain in comparison to a priori refined grids can be estimated, even for periodic, quasi-2D test case setups. Assume that a block on the coarse grid contains n_c nodes and thus has to do n_c node updates for the completion of one time step. Furthermore, an adaptively refined block is neccessarily refined in all three space directions, ending up with $n_{af} = 16 \cdot n_c$ node updates per time step: 2^3 more nodes, and twice the number of lattice time steps. If the grid was refined a priori (twice the resolution, but the same resolution in the direction of periodicity), $n_f = 8 \cdot n_c$ node updates would be sufficient. The break-even point for the refined block portion ξ can be estimated by balancing these node update quantities:

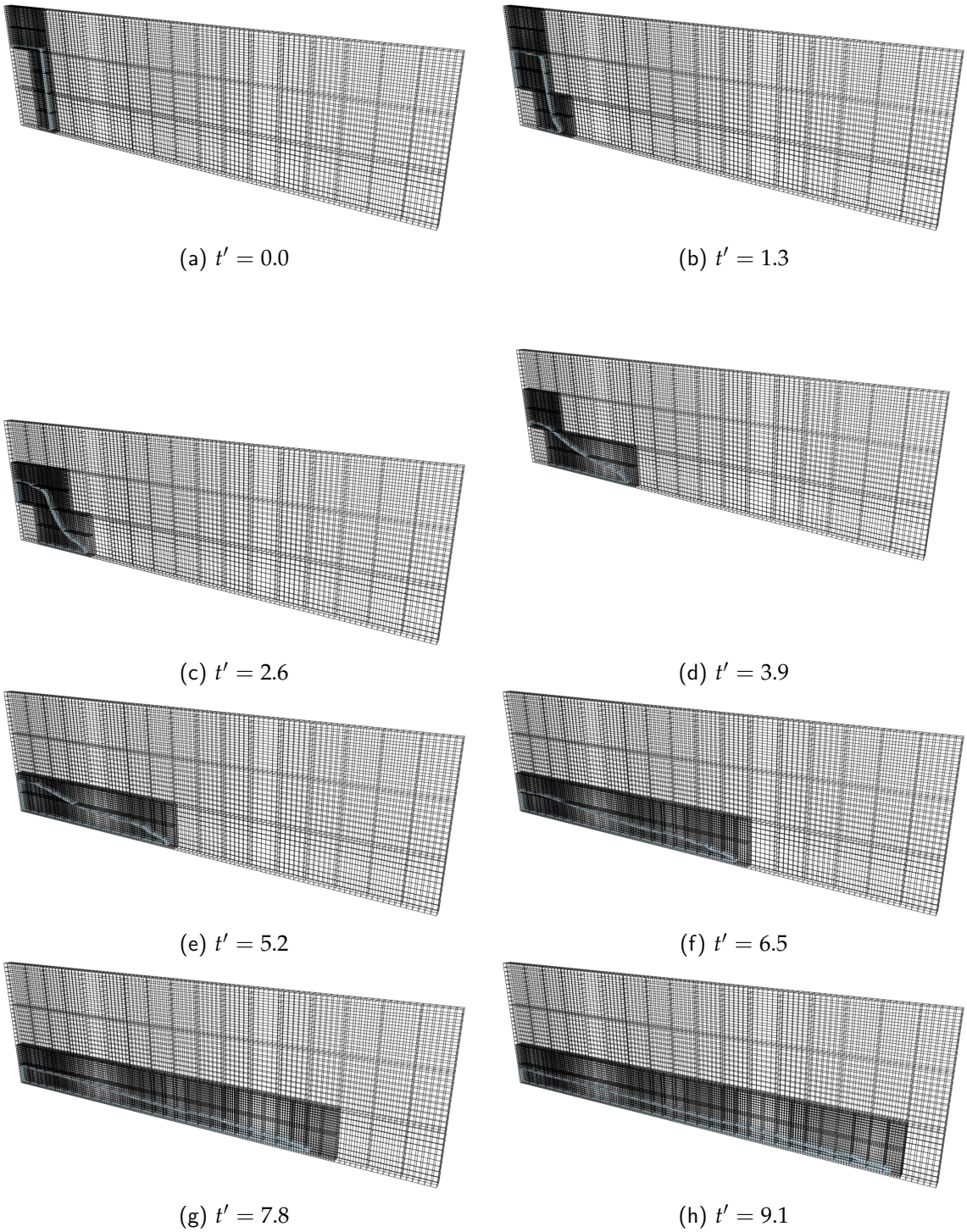
$$(1 - \xi) \cdot n_c + \xi \cdot n_{af} = \cdot n_f \tag{7.10}$$

$$(1 - \xi) \cdot n_c + \xi \cdot 16 \cdot n_c = 8 \cdot n_c \tag{7.11}$$

yielding the solution for the critical ratio ξ_c to

$$\xi_c = \frac{7}{15}. \tag{7.12}$$

For refined grids with less than 46.67% of refined blocks, the adaptive block refinement even pays off for the quasi-2D, periodic case. As a typical application for adaptively refined grids, the dambreak phenomenon is presented in Fig. 7.13. The ratio of refined blocks increases from initially $4/51 \approx 8\%$ to $16/51 \approx 31\%$. Similar to the GPU implementation on uniform grids, again a relation between the interface evolution and the performance can be seen.



(a) $t' = 0.0$ (b) $t' = 1.3$

(c) $t' = 2.6$ (d) $t' = 3.9$

(e) $t' = 5.2$ (f) $t' = 6.5$

(g) $t' = 7.8$ (h) $t' = 9.1$

Figure 7.13: Adaptive dam breaking, selected time steps (dimensionless time $t' = t\sqrt{g/h}$)

CHAPTER 8

Coupling to Numerical Wave Tanks

After the initial validation of the hybrid LB-VF model, more complex free surface flow simulations shall be adressed. For this purpose, a coupling to a numerical wave tank on the basis of the Fully Nonlinear Potential Flow (FNPF) is established.

The wave tank used in this work has been developed at the Department of Ocean Engineering of the University of Rhode Island (URI). In past work, Grilli et al. performed numerical simulations in two- (Grilli and Subramanya [63]) and three-dimensional (Grilli et al. [62]) models based on Fully Nonlinear Potential Flow (FNPF) theory. These models were used to simulate strongly nonlinear wave generation, propagation and shoaling, up to wave overturning, as well as wave structure interactions (i.e., induced motion and forces), and dissipation through breaking or absorption in a numerical beach. These processes take place in the field as well as in laboratory tanks, which has led such models to be referred to as Numerical Wave Tanks (NWTs). The main limitation of FNPF models is that waves cannot be calculated beyond overturning and impact of a breaker jet on the free surface. To do so, models with more complete physics are required. In past work, Grilli et al. coupled their FNPF-NWTs to Navier-Stokes (NS) models (with Volume of Fluid (VOF) interface representation) solving fully turbulent flows (e.g., Guignard et al. [67]; Biausser et al. [11]), thanks to various turbulence models (e.g., LES; Harris and Grilli [74]). In a first type of approach, separate inviscid and viscous domains were used and coupling was achieved through boundary conditions. In a second type of approach, a perturbation method, separating flows into inviscid plus viscous perturbation parts, was used and the solution was carried out in overlapping FNPF and NS domains. In the latter, only the perturbation flows were solved for, which led to very natural homogeneous far-field radiation conditions for these.

In this chapter, we study both coupling approaches mentioned above (i.e., separate or overlapping domains), between a 2D-FNPF-NWT and our 3D-LBM free surface model. However, coupling between a continuum mechanics-based model such as the FNPF-NWT and the kinetic LBM is less straightforward than the earlier FNPF-NS coupling. In particular, this requires developing new volumetric forcing terms or alternative collision operators for the LBM, representing wave-induced momentum

input. After a brief presentation of the underlying numerical method for the numerical wave tank (FNPF) equations for achieving the NWT-LBM coupling are developed. Finally, to validate our initial implementation, we present an application to the classical solitary wave shoaling and breaking over a plane slope.

8.1 Numerical wave tank

The 2D-FNPF-NWT used in this work is based on the implementation of Grilli and Subramanya [63] and Grilli and Horrillo [64]. The Laplace equation for the velocity potential is solved using a Boundary Element Method (BEM), and second-order Taylor series expansions are used, in an Eulerian-Lagrangian formulation, for the time updating of both the free surface potential and all moving boundary geometries (i.e., free surface, absorbing wavemaker). This requires solving two elliptic problems at each time step, one for the potential and one for its time derivative. Higher-order elements and very accurate numerical integration methods are used in the BEM, which make it possible to achieve extremely high accuracy of the solution and thus to perform long term simulations in the NWT without the need for smoothing or filtering of the solution. In case of long term simulations (e.g., for periodic or irregular waves), an absorbing beach, combining an "absorbing pressure" on the free surface and a lateral absorbing piston wavemaker yields negligible reflection in NWT experiments. Various ways of generating waves are available in the NWT, including flap and piston wavemakers, exact nonlinear waves (both periodic and solitary), and internal sources. Wavemakers can also be used to generate nonlinear random waves based on standard energy spectra. A feedback control loop allows to iteratively modify the wavemaker stroke spectrum to better approach the targeted spectrum.

8.1.1 NWT basics

In accordance with FNPF theory, we introduce a velocity potential $\Phi(x,y,z,t)$, which represents inviscid and irrotiational flows in such a way that the velocity is defined as the gradient of the potential $u_i^I = \nabla_i \Phi$. Hence, continuity equation becomes Laplace's equation for the potential

$$\nabla_j \left(u_i^I \right) = \nabla_j \left(\nabla_i \Phi \right) = \nabla^2 \Phi = 0. \tag{8.1}$$

Using Greens second identity, Eq. 8.1 is transformed into a boundary integral equation (BIE)

$$\alpha(\mathbf{x}_l)\Phi(\mathbf{x}_l) = \oint_\Gamma \left(G\left(\mathbf{x},\mathbf{x}_l\right) \frac{\partial \Phi\left(\mathbf{x}\right)}{\partial n} - \Phi\left(\mathbf{x}\right) \frac{\partial G\left(\mathbf{x},\mathbf{x}_l\right)}{\partial n} \right) d\Gamma. \tag{8.2}$$

with the Green's function

$$G\left(\mathbf{x},\mathbf{x}_l\right) = \frac{-1}{2\pi} \ln r, \frac{\partial G\left(\mathbf{x},\mathbf{x}_l\right)}{\partial n} = -\frac{1}{2\pi} \frac{\mathbf{r}\cdot\mathbf{n}}{r^2} \tag{8.3}$$

with $\mathbf{r} = \mathbf{x} - \mathbf{x}_l$ and $r = |\mathbf{r}|$ the distance from point $\mathbf{x} = (x,y,z)$ to a point of reference $\mathbf{x}_l = (x_l, y_l, z_l)$, both on the boundary, the outward normal vector $\mathbf{n}$, and a geometrical parameter $\alpha(\mathbf{x}_l) = \theta_l/(2\pi)$, function of the outer angle θ_l of the boundary at position x_l.

Boundary conditions

At the stationary parts of the boundary, a no-flow condition is prescribed by specifying zero velocity in the normal direction to the boundary,

$$\frac{\partial \Phi}{\partial n} = 0. \tag{8.4}$$

For wave generation using a wavemaker, the time-dependent position $\mathbf{x}_w$ and velocity $\mathbf{u}_w$ of the wavemaker are prescribed via

$$\frac{\partial \Phi}{\partial n} = \mathbf{u}_w \cdot \mathbf{n} \quad \text{and} \quad \mathbf{x} = \mathbf{x}_w \tag{8.5}$$

At the free surface boundary, the non-linear kinematic and dynamic boundary conditions are in Eulerian-Lagrangian form specified as

$$\frac{D\mathbf{R}}{Dt} = \mathbf{u} = \nabla \Phi \quad \text{and} \quad \frac{D\Phi}{Dt} = -gz + \frac{1}{2}\nabla\Phi \cdot \nabla\Phi - \frac{p_a}{\rho} \tag{8.6}$$

with free surface position $\mathbf{R}$, gravitational acceleration g, atmospheric pressure p_a, fluid density ρ and material derivative D/Dt.

Inner velocities

The solution for the velocity potential and its derivatives along the boundary is obtained in the BEM. The values for velocities at any arbitrary point inside the domain u_i^I can be explictly obtained, in a postprocessing step, as a function of the boundary solution for Φ as

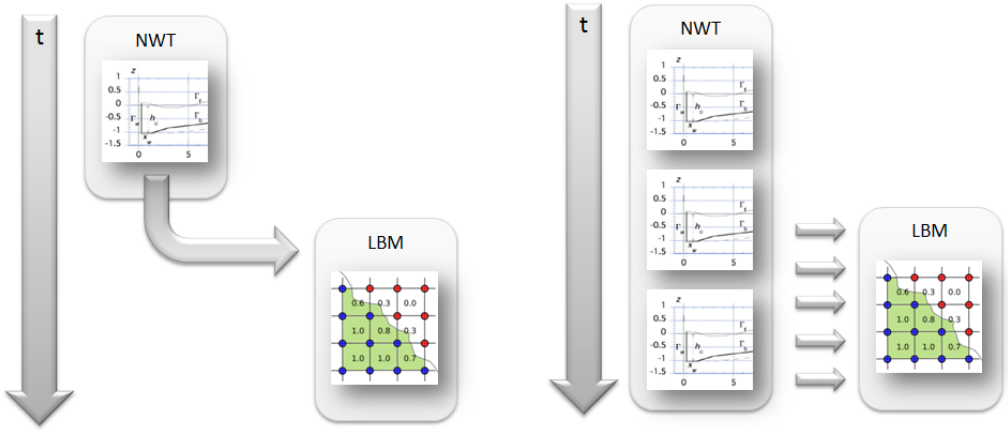
$$u_i^I(\mathbf{x}) = \oint_\Gamma \left(\nabla G\left(\mathbf{x}, \mathbf{x}_l\right) \frac{\partial \Phi\left(\mathbf{x}\right)}{\partial n} - \Phi\left(\mathbf{x}\right) \nabla \frac{\partial G\left(\mathbf{x}, \mathbf{x}_l\right)}{\partial n} \right) d\Gamma. \tag{8.7}$$

For the calculation of the velocity gradient $\nabla_i u_i^I(\mathbf{x})$, the ∇ operator is applied once more to Eq. 8.7.

8.2 NWT-LB Coupling

For complex fluid simulations, a hybrid model is desirable, combining the advantages of both FNPF and LBM frameworks. Indeed, potential flow theory is no longer applicable near rigid structures, in the surfzone or around steep bottom obstacles, where strongly nonlinear interactions, local wave breaking or significant vortex shedding and viscous dissipation occur. Moreover, the size of the viscous domain is limited, mainly due to computational expenses, and cannot meaningfully be used for the whole domain. Hence, a highly resolved and/or turbulent LBM simulation should only be performed in the vicinity of a structure or a wave breaking region, and not in the peripheral regions of the flow field.

Weak and strong coupling approaches have to be distinguished (see Fig. 8.1): in a weakly coupled approach, only an initial transfer of information from the NWT to the LBM exists, whereas in a strong coupling, a continuous data exchange between NWT and LBM is typical. Both techniques will be presented in the following.



(a) Weak coupling (b) Strong coupling

Figure 8.1: Information flow in weak and strong coupling approaches

8.2.1 Weak coupling

In a weakly coupled approach the LB domain is initialized with results of the NWT. Only this initial, unidirectional coupling is specified. At the offshore fluid boundary (Fig. 8.2), stationary boundary conditions for the water height $\bar{h} = h^{NWT}(t = 0)$ and a constant inflow velocity $\bar{\mathbf{v}} = \mathbf{v}^{NWT}(t = 0)$ are prescribed.

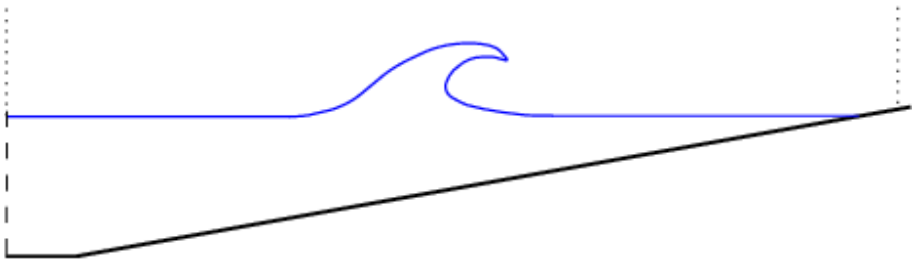


Figure 8.2: Initialization of the entire LBM domain with the inviscid, irrotational NWT solution v^I, p^I

In the LBM, based on this information, the particle distribution functions f are initialized with Maxwellian equilibrium distribution functions (Eq. 3.14)

$$f_i = f_i^{eq}(\bar{\rho}, \bar{\mathbf{v}}) \tag{8.8}$$

Alternatively, a local Poisson-type iteration might be used to further improve the non-equilibrium parts of the distribution functions, ([114], Eq. 3.35). This weakly coupled approach requires to model the total flow in the LBM. Moreover, the size of the LB domain must be large enough to avoid perturbations from the stationary boundary condition.

Weak coupling with transient boundary conditions

In order to reduce the domain size, transient boundary conditions may be used (Fig. 8.3) and time-dependent values for velocity and position of the free surface can be prescribed on the leftward boundary ($\bar{\mathbf{v}}(t) = \mathbf{v}^{NWT}(t)$).

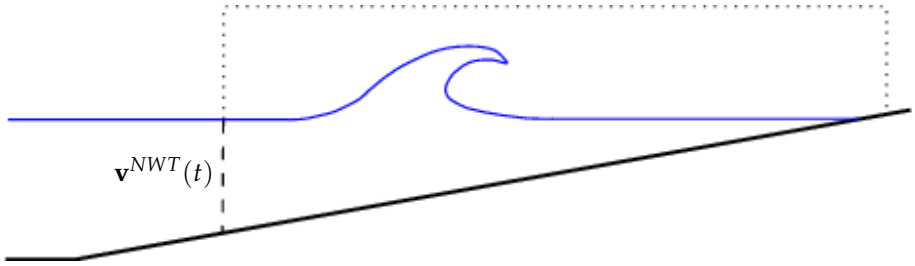


Figure 8.3: Weak coupling of LBM and NWT with transient boundary conditions

8.2.2 Strong coupling

In the strongly coupled approach, the NWT does not only serve to initialize the LB computations, but also drives computations either via transient boundary conditions, or via volumetric source terms in the momentum equation. In both cases the domain size can be reduced significantly (Fig. 8.4).

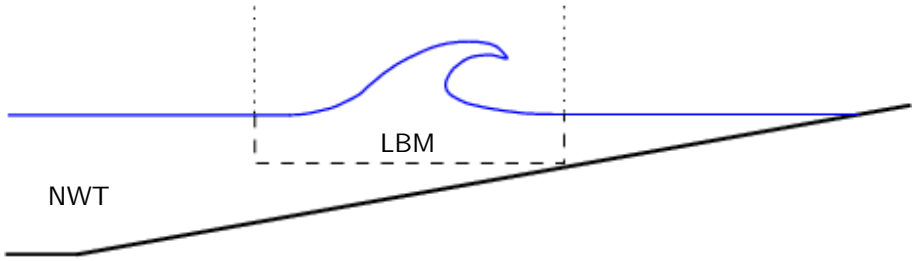


Figure 8.4: Strong coupling of LBM and NWT via a perturbation approach

For wave-induced flows, the viscous perturbation caused by a structure, bottom geometry, or a beach onto the otherwise nearly inviscid, irrotational flow is expressed explicitly in the model. As indicated before, pressure and velocity field are split up into the irrotational, inviscid part $p^I, \mathbf{v}^I$ and the rotational, viscous perturbation $p^P, \mathbf{v}^P$

$$\mathbf{v} = \mathbf{v}^I + \mathbf{v}^P \quad \text{and} \quad p = p^I + p^P. \tag{8.9}$$

with $\mathbf{v}^I$ obtained from the NWT, while the LBM solves for $\mathbf{v}^P$. In the remainder of the derivation, we switch to indicial notation ($\mathbf{v} = v_i$), so that the Navier-Stokes equation read:

$$\frac{\partial v_i}{\partial t} + v_j \frac{\partial v_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + (\nu + \nu_T)\frac{\partial^2 v_i}{\partial x_j^2} + \frac{\partial \nu_T}{\partial x_j}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) \tag{8.10}$$

with velocity v_i, pressure p, viscosity ν and turbulent viscosity ν_T. The inviscid far-field wave flow is specified into the LBM via volumetric terms, which can be expressed, by analogy, by inserting Eq. 8.9 into the NS equations. After transformation, we have,

$$\begin{aligned}\frac{\partial v_i^P}{\partial t} + v_j^P \frac{\partial v_i^P}{\partial x_j} = &-\frac{1}{\rho}\frac{\partial p^P}{\partial x_i} + (\nu + \nu_T)\frac{\partial^2 v_i^P}{\partial x_j^2} + \frac{\partial \nu_T}{\partial x_j}\left(\frac{\partial v_i^P}{\partial x_j} + \frac{\partial v_j^P}{\partial x_i}\right) \\ &\underbrace{-v_i^I \frac{\partial v_i^P}{\partial x_j} - v_j^P \frac{\partial v_i^I}{\partial x_j} + \frac{\partial \nu_T}{\partial x_j}\left(\frac{\partial v_i^I}{\partial x_j} + \frac{\partial v_j^I}{\partial x_i}\right)}_{Additional\ \ terms}\end{aligned} \tag{8.11}$$

where v_i^I, the irrotational velocity field, satisfies Euler equations. One can see that, in addition to viscous and turbulent terms on the right hand side, convection-like interaction terms between v_i^I und v_i^P occur. Besides those additional source terms

$$F_i = -v_j^I \frac{\partial v_i^P}{\partial x_j} - v_j^P \frac{\partial v_i^I}{\partial x_j} + \frac{\partial \nu_T}{\partial x_j}\left(\frac{\partial v_i^I}{\partial x_j} + \frac{\partial v_j^I}{\partial x_i}\right), \tag{8.12}$$

the equation corresponds to Eq. 8.10. The velocity v_i^I and the velocity gradient are obtained from the NWT's solution at the current time step, while the perturbating part and its gradients are obtained from values of the previous time step LB solution. Alternatively, the convection-like coupling terms can be rewritten as

$$\begin{aligned} F_i &= -v_i^I \frac{\partial v_i^P}{\partial x_j} - v_j^P \frac{\partial v_i^I}{\partial x_j} && (8.13)\\ &= -v_j^I \frac{\partial v_i^P}{\partial x_j} - v_j^P \frac{\partial v_i^I}{\partial x_j} - v_i^I \frac{\partial v_j^P}{\partial x_j} - v_i^P \frac{\partial v_j^I}{\partial x_j} && (8.14)\\ &= -\frac{\partial}{\partial x_j}\left[v_i^I v_j^P + v_i^P v_j^I\right], && (8.15) \end{aligned}$$

which is valid for divergence-free velocity fields v_i^I and v_i^P with $v_{j,j}^I = 0$ and $v_{j,j}^P = 0$. Hence, the coupling terms are included in the momentum flux tensor directly, yielding

$$\frac{\partial v_i^P}{\partial t} + \frac{\partial}{\partial x_j}\left[v_i^P v_j^P + v_i^I v_j^P + v_i^P v_j^I\right] = -\frac{1}{\rho}\frac{\partial p^P}{\partial x_i} + (\nu + \nu_T)\frac{\partial^2 v_i^P}{\partial x_j^2} + \frac{\partial \nu_T}{\partial x_j}\left(\frac{\partial v_i^P}{\partial x_j} + \frac{\partial v_j^P}{\partial x_i}\right) \tag{8.16}$$

These additional terms can be incorporated in a modified MRT collision operator Ω with modified equilibrium moments for the momentum advection [151]. They read

$$\begin{aligned} m_1^{eq} &= e^{eq} = \rho_0\,(v_x^2 + v_y^2 + v_z^2 + 2\,v_x v_x^I + 2\,v_y v_y^I + 2\,v_z v_z^I), && (8.17a)\\ m_9^{eq} &= 3p_{xx}^{eq} = \rho_0\,(2\,v_x^2 - v_y^2 - v_z^2 + 4\,v_x v_x^I - 2\,v_y v_y^I - 2\,u_z u_z^I), && (8.17b)\\ m_{11}^{eq} &= p_{zz}^{eq} = \rho_0\,(u_y^2 - u_z^2 + 2\,v_y v_y^I - 2\,v_z v_z^I), && (8.17c)\\ m_{13}^{eq} &= p_{xy}^{eq} = \rho_0\,(u_x\,u_y + v_x v_y^I + v_y v_x^I), && (8.17d)\\ m_{14}^{eq} &= p_{yz}^{eq} = \rho_0\,(u_y\,u_z + v_y v_z^I + v_z v_y^I), && (8.17e)\\ m_{15}^{eq} &= p_{xz}^{eq} = \rho_0\,(u_x\,u_z + v_x v_z^I + v_z v_x^I). && (8.17f) \end{aligned}$$

Note that, as a basis for the derivation, incompressibility has been assumed to get rid of the compressibility correction terms. Hence, we expect a high Mach number dependancy of this collision operator, which will be checked in the following.

8.2.3 Parametrization

In order to transfer the simulation results from the NWT to the LB model, a parametrization has to be established, to select the LB parameters for grid spacing Δx, Mach number Ma, forcing g_{LB} and viscosity ν. This involves three steps.

First, as the solutions of the LBM satisfies NS equations, up to an order $\mathcal{O}(Ma^2)$, it is indispensable to observe and prescribe the maximum Mach number Ma and hence the maximum velocity $v_{max} = \mathrm{Ma} \cdot c_s = \mathrm{Ma} \cdot \frac{c}{\sqrt{3}}$ in the LB simulation. Second, free surface flows are qualified by their Froude number, i.e., a dimensionless number that compares inertia and gravitational forces, $Fr = v / \left(gh\right)^{0.5}$ using maximum velocity v, gravity g and water depth h. The Froude numbers of the NWT and the LBM must be identical, so that based on a given LB discretization we can calculate the LB gravitational term $g_{LB} = v_{max}^2 \mathrm{Fr}^{-2} h_{LB}^{-1}$. Third, the Reynolds number of experiments and numerical simulations should be the same. The NWT is based on inviscid potential flow theory, so that a Reynolds number cannot be assigned. Nonetheless we can calculate a corresponding Reynolds number via $Re = \frac{vh}{\nu_{water}}$ and, consequently, find the resulting LB viscosity as $\nu_{LB} = v_{LB} h_{LB} \mathrm{Re}^{-1}$

Finally, we find that NWT results for velocity and pressure should be transferred to the LB simulations by applying the following scaling factors

$$\Delta v = \frac{v_{max,LB}}{v_{max,NWT}} \quad \text{and} \quad \Delta x = \frac{h_{LB}}{h_{NWT}} \tag{8.18}$$

$$\Delta t = \frac{\Delta x}{\Delta v} \quad \text{and} \quad \Delta p = \frac{\rho_{LB} g_{LB} h_{LB}}{\rho_{NWT} g_{NWT} h_{NWT}} \tag{8.19}$$

It turns out that the only free parameters in these equations are the grid resolution (which directly governs Δx) and the Mach number limit ($\mathrm{Ma}_{max} = 0.1$ is considered a reasonable maximum value for an incompressible limit).

8.3 Validation of the weak coupling

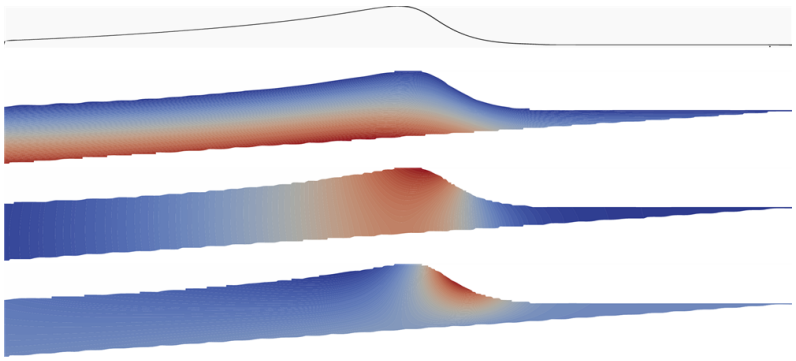
The weak coupling of NWT and LBM in fact is a straightforward initialization of the LB domain, which is usually used in most transient CFD applications to start with valid initial conditions. Apart from the pure initialization step, a proper selection of boundary conditions and the correct parametrization are crucial. The application of the weakly coupled algorithm is illustrated for the case of a solitary wave that breaks during shoaling, which has been analyzed on consecutively refined LB grids. This test case is extremely demanding in terms of the free surface capturing scheme, as already has been observed by several other VOF methods (Guignard et al. [67], Lachaume et al. [100], Biausser et al. [11]). The results of the NWT simulations for the initialization of water height, velocity and pressure fields are shown in test case setup 8.1 for a solitary wave of height $H = 0.5m$ over a 1:15 slope.

The simulations are run for four different grid resolutions and two Mach numbers, as given in test case setup 8.1. The general expected flow behaviour is well represented, and the breaker jet agrees very well with results of purely potential flow simulations (e.g., [65]) as illustrated in Fig. 8.5 for time $t = 3.0s$.

The test case was challenging and served as the ultimate validation example during the development of the advection scheme. Even though more simple validations as the breaking dam setup worked fine even with less sophisticated free surface capturing methods, for the breaking wave, a high-end model was mandatory. At the free surface boundary, a linear extrapolation of surface velocities for the anti-bounce-back pressure boundary condition is used. The flux calculation is extended to eighteen directions instead of the previous, simpler six-face model, so that diagonal fluxes are represented accurately as well.

Param.	Value
Domain	12m x 1.40m
Lattice	$300 \times 1 \times 45$
	$600 \times 1 \times 90$
	$1200 \times 1 \times 180$
	$1800 \times 1 \times 270$
Re	$\approx 630,000$
Fr	≈ 0.84
Ma	0.01,0.001
g	$9.81\mathrm{m/s^2}$
ν	$1 \cdot 10^{-6}\mathrm{m^2/s}$

(a) Parameters



(b) Initialization (Interface position, ρ, $vx1$, $vx3$)

Test case 8.1: Breaking wave during shoaling

Finally, the results of our hybrid model are of high quality compared to other VOF free surface capturing codes. Note that - even if the test case is two-dimensional - the simulation has been carried out with the full three-dimensional solver using periodic boundary conditions in y direction. Hence, the simulation of three-dimensional plunging breakers or even freak waves is possible without further modifications of the code.

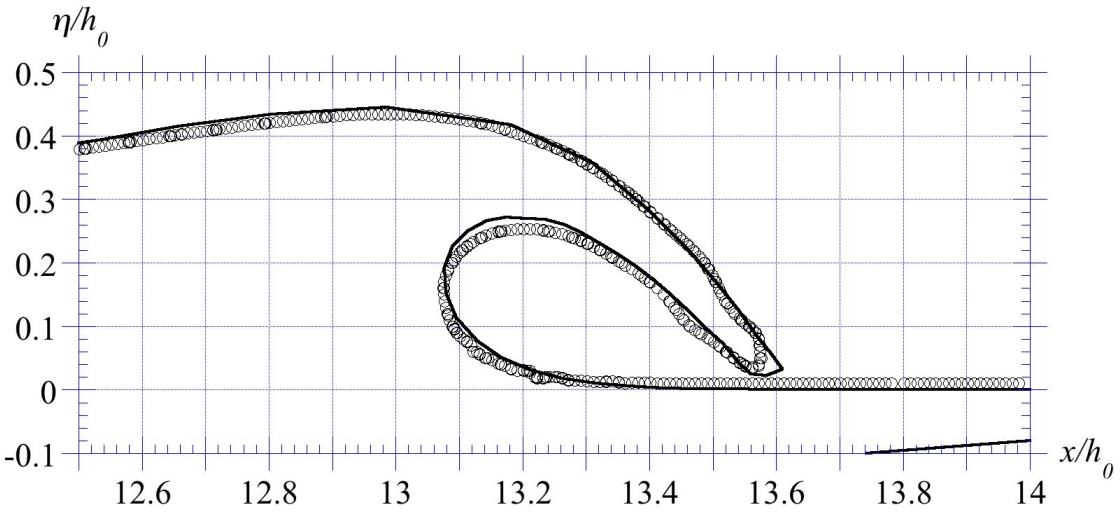
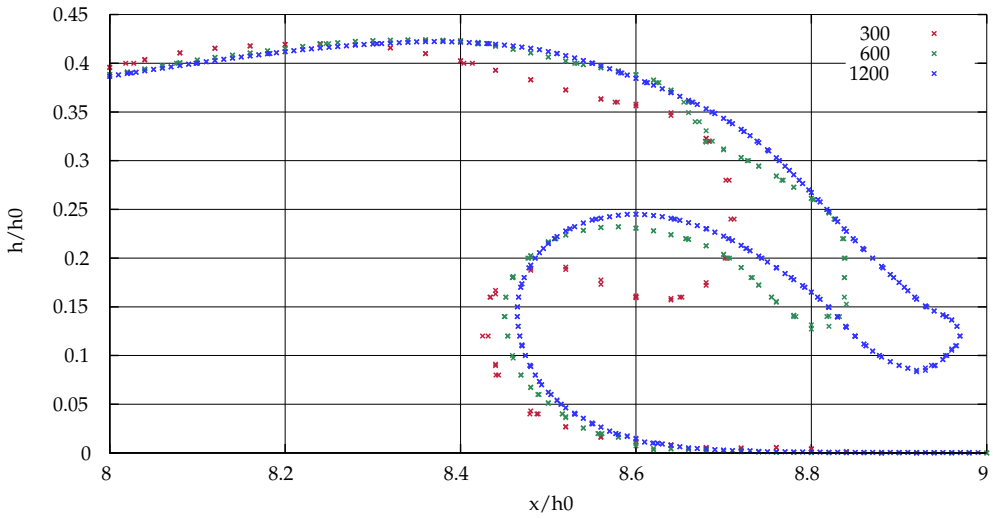
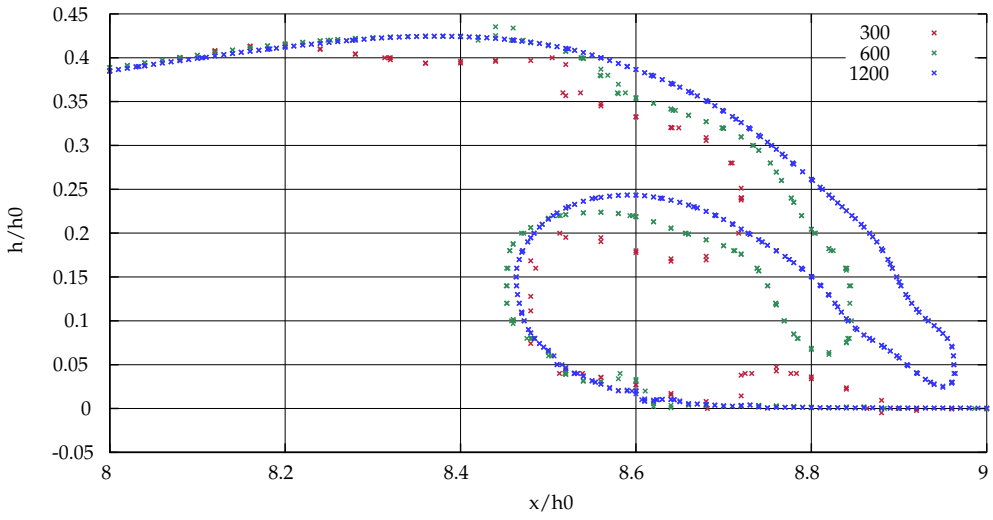


Figure 8.5: Comparison of LBM-VOF model to the FNPF results of Grilli et al. [65]



(a) Ma 0.01



(b) Ma 0.001

Figure 8.6: Results for two different Mach numbers

8.4 Validation of the strong coupling

In the weak coupling, the critical aspects were identified to be the free surface capturing scheme. Opposite to that, for the strong coupling, the coupling formulation itself is part of the validations, and merely test cases that do not involve free surfaces will be examined. For strongly coupled test cases including free surfaces, more aspects such as the dynamic free surface boundary condition have to be considered, what is beyond the scope of this work. Once the general suitability of the hybrid, strongly coupled approach for bulk flow problems has been shown, these extensions will be part of further research.

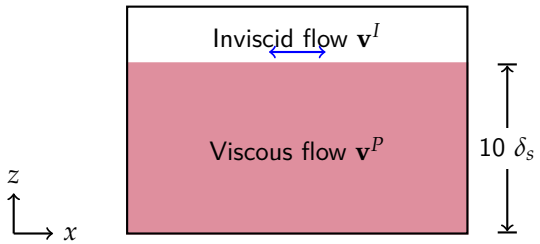
Initially, a laminar oscillatory boundary layer is examined. Secondly, the Eulerian drift in a wave-induced boundary layer serves to qualify and quantify the coupling in detail, comparing both coupling approaches.

8.4.1 Laminar oscillatory boundary layer

The test case serves to compute the velocities in laminar oscillatory boundary layers. The solution of the invisvid far-field is known to be $u^I = u_0 \sin(\omega t)$ with $\mathbf{v} = (u, v, w)$, and does not include viscous effects in the boundary layer. The inviscid velocity is applied as boundary condition at the top boundary (upper z- boundary) of the domain, whereas at the bottom, a no-slip boundary condition is imposed. For x and y direction periodic boundary conditions are used.

Parameter	Value
Lattice	4 x 4 x 128
Re	10
Ma	0.1
δ_s^{LB}	12.8

(a) Parameters



(b) Geometry

Test case 8.2: Laminar oscillatory boundary layer

We solve for the perturbating part $\mathbf{v}^P, p^P$ of the flow field only, so that the boundary conditions have to be inverted. The viscous perturbation has to vanish at the top boundary so that the overall fluid velocity equals the inviscid velocity u^I. At the bottom boundary, vice versa, it has to balance the viscous part to fulfill the no-slip boundary condition:

$$u^P(y = 0) = -u_0 \sin(\omega t) \quad \text{and} \quad u^P(y = 10\delta_s) = 0 \tag{8.20}$$

The final solution for u is obtained by superposition of the inviscid and perturbating parts:

$$u(y, t) = u^I + u^P = u_0 \sin(\omega t) + u^P \tag{8.21}$$

The resulting velocity profiles for four different phase angles in comparison to the analytical solution

$$u_a = u_0 \left(\sin(\omega t) - e^{-\frac{y}{\delta s}} \sin\left(\omega t - \frac{y}{\delta s}\right) \right) \tag{8.22}$$

are given in Fig. 8.7. Very good agreement can be seen. Moreover it can be deduced that the strong coupling approach does not influence the quality of the results. For the forcing-based approach, this is obvious, as $u = u(y)$ and both other velocity components equal zero and the coupling terms vanish. Similar, the use of the modified collision operator did not affect the results neither, which is as expected: most of the interaction terms in Eq. 8.17 - Eq. 8.17f vanish due to $v^I = w^I = 0.0$. The remaining velocity component does not influence the result.

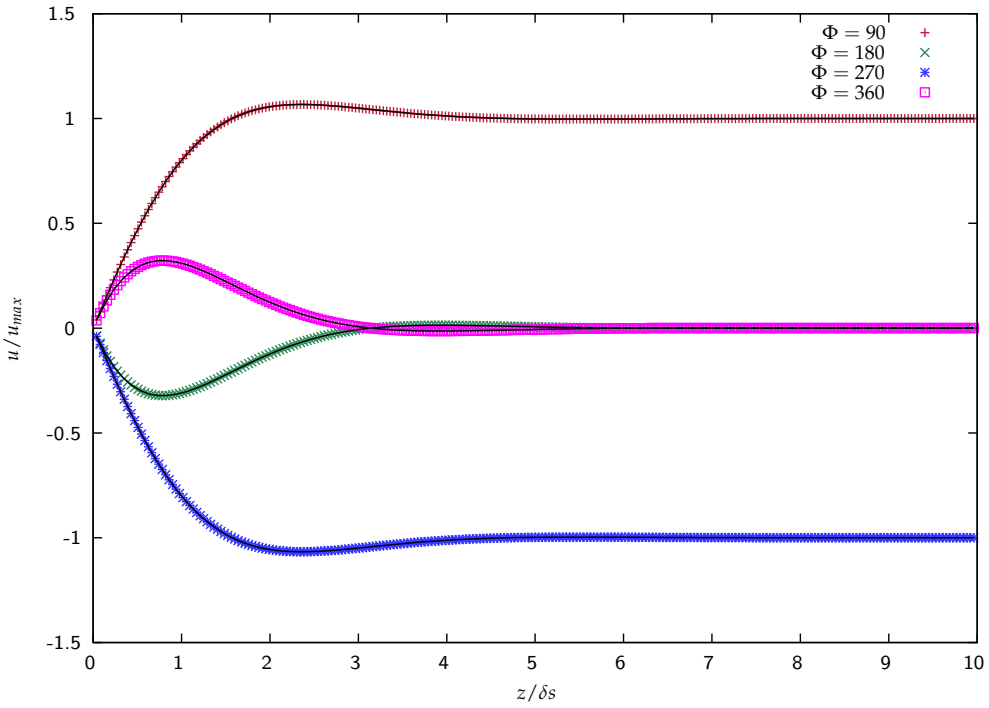


Figure 8.7: Velocity profiles for the laminar oscillatory boundary layer, in comparison to the analytical predictions

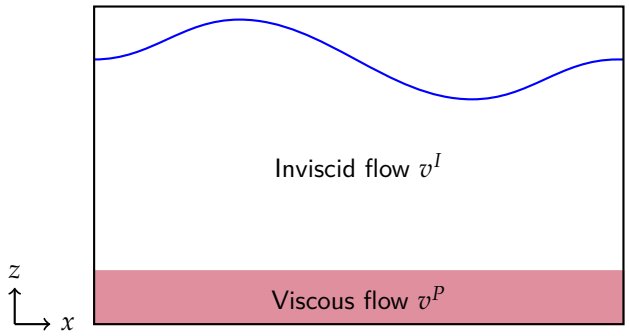
8.4.2 Wave-induced boundary layer

For the validation of the coupling terms, the wave-induced boundary layer below progressive waves serves as a more difficult and demanding test case. A strong coupling by means of the convection-like forcing terms (Eq. 8.12) failed for this test case. The finite-difference approximation of the velocity gradient terms in combination with the explicit time-stepping scheme induced instabilities. For future work, the use velocity gradient information which is locally available at each lattice node possibly stabilizes the coupling forcing terms. In the following, the modified collision operator serves to drive the LB computation of the perturbation fields.

According to linear or higher-order wave theories, the velocity and pressure field below progressive waves are periodic in time, and integrate to zero. Nonetheless, a small but significant mass transport can be observerd in the boundary layer below progressive waves. Longuet-Higgins [108] was the first to show the occurrence of this steady streaming in the oscillatory boundary layer under progressive waves. Harris and Grilli [74] have shown that the steady streaming even occurs when the inviscid velocity field is obtained from linear wave theory. At the boundary, we demand no-slip at the bottom,

Param.	Value
A	$0.01\ m$
g	$9.81\ m/s^2$
k	$12.566\ m^{-1}$
ω	$10.472\ s^{-1}$
h	$0.113\ m$
ν	$5.11 \cdot 10^{-5} m^2/s$
Domain	$\frac{2\pi}{k} = 0.5m$ x $16 \cdot \delta_s = 0.05m$
Lattice	10x100,20x200,40x400,80x800

(a) Parameters



(b) Geometry

Test case 8.3: Wave-induced boundary layer

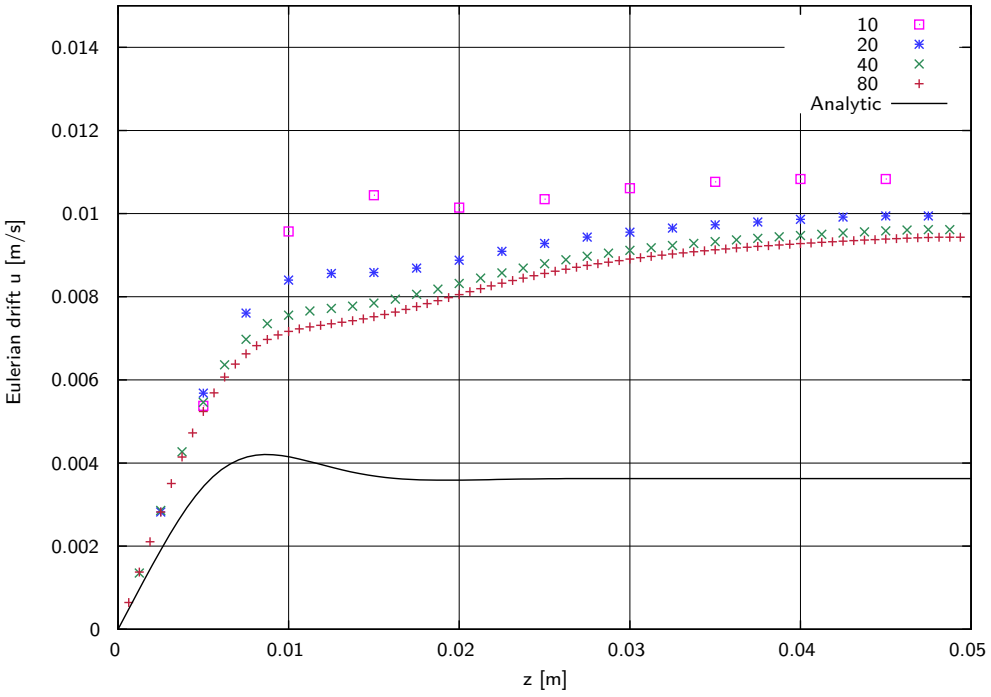
i.e. zero total velocity. At the top boundary, we assume that the viscous perturbation does not change with respect to the wall normal direction, i.e. we assume zero gradient for all velocity components. The resulting boundary conditions for the LB solution for the perturbation fields read

$$\mathbf{v}^P(z=0) = -\mathbf{v}^I(z=0) \quad \text{and} \quad \partial_z \mathbf{v}^P(y=16\delta_s) = \mathbf{0} \tag{8.23}$$

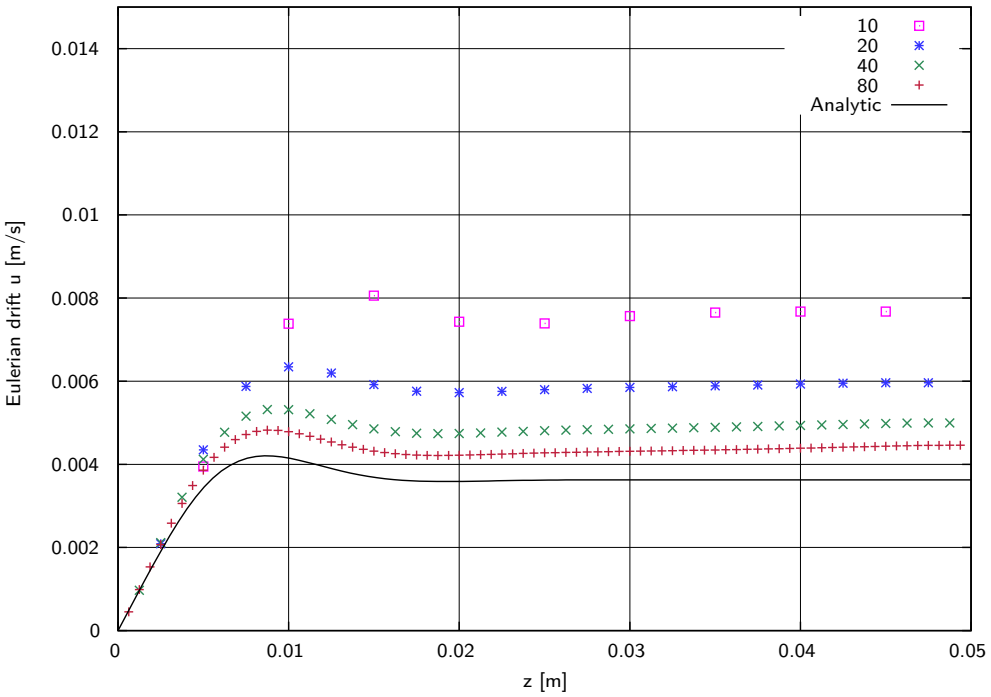
The mean Eulerian drift in the wave-induced boundary layer is compared to the analytical solution given by

$$u = \frac{A^2\omega k}{\sinh^2 kh}\left[\frac{3}{4} - e^{-\xi}\cos\xi + \frac{1}{2}e^{-\xi}\sin\xi + \frac{1}{4}e^{-2\xi} - \frac{1}{2}\xi e^{-\xi}\cos\xi - \frac{1}{2}\xi e^{-\xi}\sin\xi\right]. \tag{8.24}$$

In Fig. 8.8 the results for two distinct Mach Numbers Ma $= 0.01$ and Ma $= 0.001$ and four grid resolutions are given. Convergence to the analytical solution can be observed, whereas the Mach number dependancy is dominating, compared to the discretization error in terms of grid size Δx. Nonetheless it can be seen that the collision operator in general is applicable to strongly coupled simulations of wave-induced, viscous effects. The limitation in terms of Mach number is a severe limiting factor at first sight. However, recalling that for the simulation of the breaking wave during shoaling a Mach number of around 0.001 was necessary too, this constraint does not affect the overall algorithm.



(a) Ma 0.01



(b) Ma 0.001

Figure 8.8: Mean Eulerian drift in the wave-induced boundary layer, results for two selected Mach numbers and four consecutively refined grids

CHAPTER 9

Free Surface Flow and Fluid-Structure Interaction

Fluid-Structure interaction plays an important role in the field of civil engineering and has been part of extensive research since years. The realistic transient simulation of fluid structure interaction processes is a challenging task, both for the numerical techniques and the required computer capacities behind. Apart from the algorithmic challenge to couple different code frameworks, the interaction introduces complex numerical and mechanical additional phenomena which require special care.

The main challenge remains the coupling of both structural and fluid solvers, which is somewhat different for different discretization schemes. In general, two approaches to solve fluid-structure interaction (FSI) problems exist: monolithic approaches use the same discretization schemes for both the governing fluid and solid equations. The coupling conditions at the fluid- structure interface are fulfilled in a strong sense, and the time steps of the two solvers and the meshes of solid and structure have to be matching. For further details see [85, 169, 84]. Opposite to that, partitioned approaches use separate solvers for the fluid and the structural parts of the system. Communication between the solvers is mandatory in order to exchange information on the physical properties at the fluid-structure interface, and to fulfil the coupling conditions. In non-monolithic approaches, two different numerical formulations for the two domains are used. For both subparts, the most efficient and optimal solvers may be used, and time steps and grid sizes can be fine-tuned to obtain optimal results in both subdomains. For the information exchange, explicit and implicit information exchange patterns have been realized.

Concerning the numerical simulation of hybrid FSI and free surface flow problems, a lot of research has been done in the MARIN group in the Netherlands. Kleefsman [94] examines the effects on green water loading on ship decks, Wemmenhove et al. [174] aims at tank sloshing phenomena, with finite-element modeling of the container side walls. Recently, Abadie et al. [1] proposed a three-fluid approach to simulate wave generation by Russel- and Heinrich-type wavemakers, and in the long run, wave generation by landslides shall be adressed.

9.1 Structural models

For the simulation of the structural part, various kinds of solvers can potentially be coupled to the LB fluid solver. Full three-dimensional frameworks on the basis of the finite element method (FEM) provide higher-order solutions on the basis of a non-linear description of geometry and non-linear material laws, including effects of plastification and structural damage. In contrast to that, rigid body engines neglect the deformation of the solids but focus on their collisions, friction and the interaction of a large number of rigid bodies. For the hybrid FSI-free surface simulations, which are presented in this chapter, solely the *PE physics engine* is used for the simulation of the structural part. The PE is developed and maintained by Iglberger et al. [86] at the University of Erlangen-Nuremberg. In this particle-based framework, rigid body motion on the basis of Newtons second law is modeled. Elastic deformations of the rigid bodies are not considered. Contacts and collisions between the rigid bodies are treated on the basis of relative velocities between them. The resulting linear complementary problem (LCP) can be solved with several solvers, e.g. a projected Gauß-Seidel method. Recently, an algebraic multigrid solver has been added to the PE by Iglberger to allow for massively parallel simulations with the PE. This is desirable, especially when focusing on simulations with hundreds or thousands of particles. So far, three geometric primitives (sphere, cuboid and capsulus) are available for the solid body modeling. Results of a typical simulation with the PE are shown in Fig. 9.1. This test case is taken from the PE tutorial and demonstrates the impact of a sphere on a box stack. Single boxes are kicked out of the stack and finally, the box stack collapses under the influence of gravity. In the long run, the coupling to a higher-order finite element solver

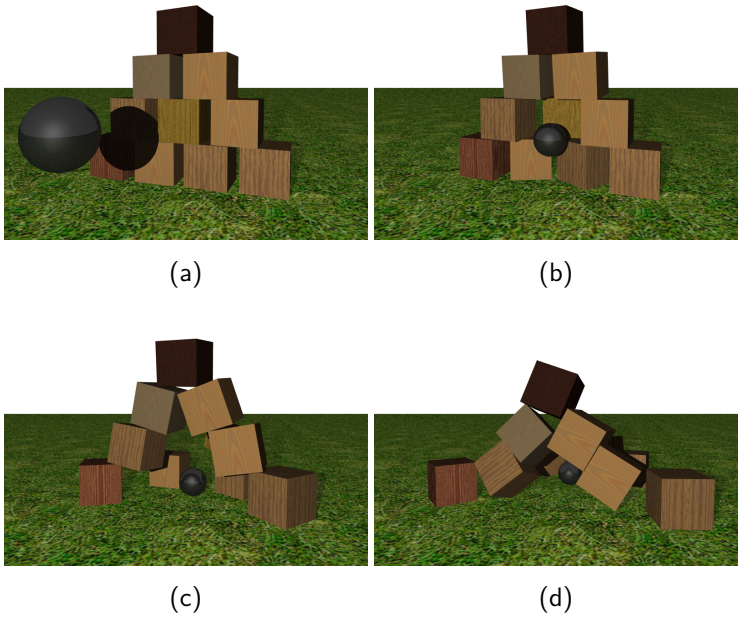


Figure 9.1: Box stack (PE only)

(which already has been established for onephase flow problems by Geller [49]) will be adapted for free surface flow simulations. However, in this chapter, the basic extension of the free surface scheme to deal with moving solid objects in the flow is in the center of attention.

9.2 Coupling approach

For a coupling to the explicit LB method, a non-monolithic coupling approach is necessary. We use a bidirectional, explicit coupling approach (see Fig. 9.2). The fluid force acting on the rigid bodies is evaluated by means of the momentum exchange method [101]. Since the rigid bodies do not allow elastic or plastic deformations, the evaluation of the integral force on the whole rigid body is sufficient and the stress integration method [50] does not have to be used. After the data transfer to the PE, the calculated force is applied to the rigid bodies and one time step of rigid body motion is computed. The resulting displacements and velocities are passed back to the fluid solver, where the geometry is updated, and a modified second-order bounce back scheme serves to set the corresponding rigid body boundary velocity [13].

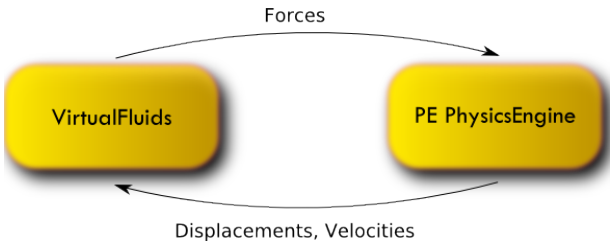


Figure 9.2: Bidirectional coupling approach

The explicit, bidirectional coupling approach has been validated in several publications. Bettah et al. [9] examined soil liquefaction under the influence of varying fluid pressures. The granular soil material is modeled with the PE phyics engine, the fluid is modeled in VirtualFluids. Geller [49] used a similar explicit and bidirectional coupling approach for advanced FSI simulations with a higher-order p-FEM solver, and validated it against experimental benchmark data of the FSI community. In any case, the explicit LB coupling approach has shown to be very competitive, in terms of accuracy and computational time.

9.3 Extension of the free surface model

For the free surface model, only minor modifications are necessary to incorporate moving objects. Basically, the same concepts as for the treatment of impermeable solid boundaries are used:

- The free and non-solid volume $1 - \zeta$ of each interface cell plays a central role in the VOF method and has to be known.
- The modified Parker-Youngs approximation on the basis of asymmetric finite difference stencils is used for the evaluation of the surface normal next so solid boundaries.
- The flux terms for the flux between two boundary cells are weighted. The mean solid fraction of each pair of cells is used as correction factor.

Hence, the only extension to the pure free surface algorithm (chapter 7) is that the information on the solid volume in each cell has to be renewed in every time step after the rigid bodies are advected. Moreover, the velocity terms for the second-order bounce-back scheme are updated in every time step.

9.4 Validation

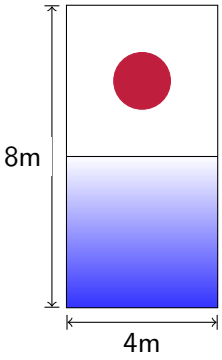
In the following, two straightforward validation examples for the partitioned coupling approach in a free surface flow context are given. The falling sphere demonstrates that the algorithm is able to handle large topological changes, such as solid objects that penetrate a closed water surface. Secondly, the simulation of a Russel wave generator serves to validate the overall model against experimental data, which was recorded in an experimental wave tank.

9.4.1 Equilibrium position of bouyant objects

In this test case, a solid sphere is positioned in a certain height above a water basin with a flat surface at rest. The sphere then is released and accelerates towards the water surface. As the sphere penetrates the surface, its movement is damped due to bouyancy effects and additional contributions of the dymamic pressure gradient, i.e. drag. In an ideal fluid, this would lead to infinitely long oscillations, as the total energy is conserved. Opposite to that, the fluid is assumed to be viscous in our LB model, so that the sphere movement is damped, and the position of the sphere finally converges to a static swimming position, which is depending on the ratio of sphere density to fluid density. The simulation parameters are given in test case setup 9.1.

Parameter	Value
Domain	4m x 4m x 8m
Sphere	$R = 0.75,\ \mathcal{C} = (6.0, 2.0)$
	$\rho_s = 500, 600, 700, 800, 900 kg/m^3$
ν	$1 \cdot 10^{-6} \mathrm{m}^2/\mathrm{s}$
g	$9.81 \mathrm{m}/\mathrm{s}^2$
Lattice	32 x 32 x 64

(a) Parameters



(b) Geometry

Test case 9.1: Immersing sphere

The height h of the non-wetted part of the cap of a sphere (radius r, density ρ_S) which is swimming in a fluid (density ρ_W) may be calculated via solving

$$1 - \frac{h^2(3r - h)}{4r^3} = \frac{\rho_S}{\rho_W} \tag{9.1}$$

for h. The results for five different sphere densities are given in Fig. 9.4. The theoretical values for water height h_w and sphere penetration Δ are compared to the simulation values h_w^{sim}, Δ^{sim} and the final results for the sphere centroid h_c^{sim}. The relative error in penetration depth Δ in relation to the sphere diameter D is below 1 % for all five simulations. Apart from that, the coupling algorithm introduces a slight mass loss, leading to a relative error in terms of the final water height ε_h of 3-4%.

In Fig. 9.3, the vertical coordinate of the sphere centroid is plotted over time for spheres of densities 500, 600 and 700 kg/m^3. Before the spheres touch the water surface, they share the same path,

as the terminal velocity is independant on the density, at least for the frictionless case. After the penetration, the oscillatory behaviour is damped. The higher the density, the higher the influence of sphere inertia, the slower the decay of oscillations. Finally, for all three cases, the rigid bodies arrive at their static swimming position h. In the plot, the dashed line indicates the target position of the sphere centroid on the basis of the present final simulation water level h_w^{sim}.

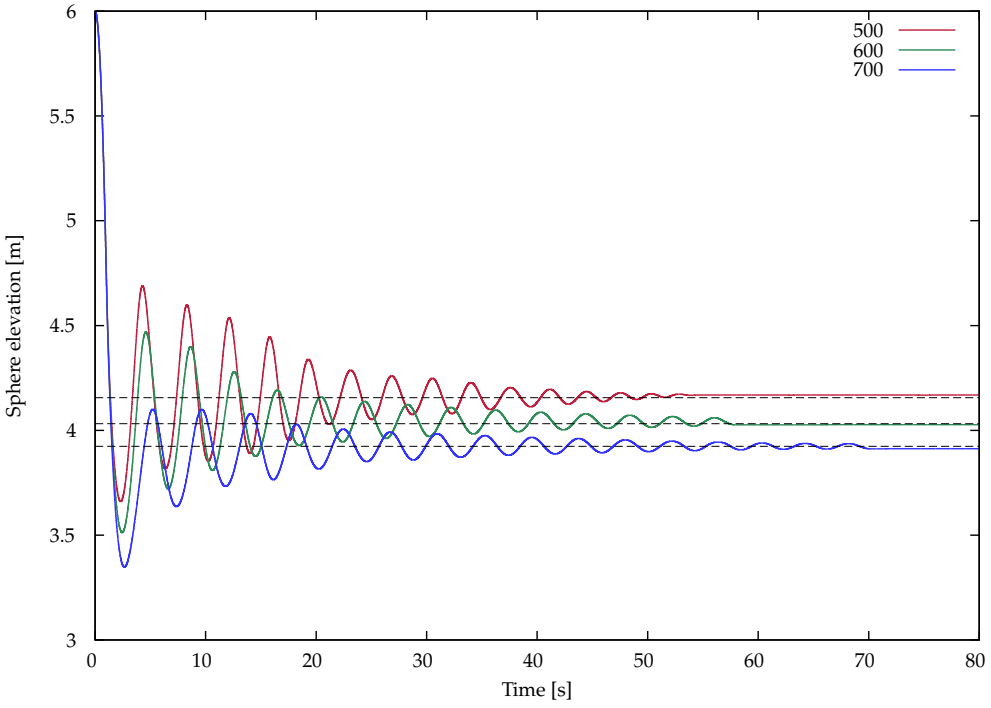
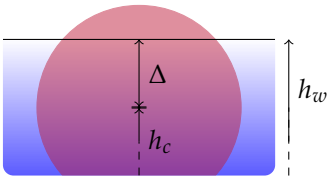


Figure 9.3: Displacement of the sphere centroid over time, for three different sphere densities and in comparison to the target value (dashed line)

In Fig. 9.5, selected time steps for the case with sphere density $\rho_s = 900\text{kg/m}^3$ are given during the initial stage of the simulation before and during the first impact. The sphere submerges and is temporarily completely underwater, due to the high inertia forces, before the restoring buoyancy force leads to a reverse acceleration towards the water surface.

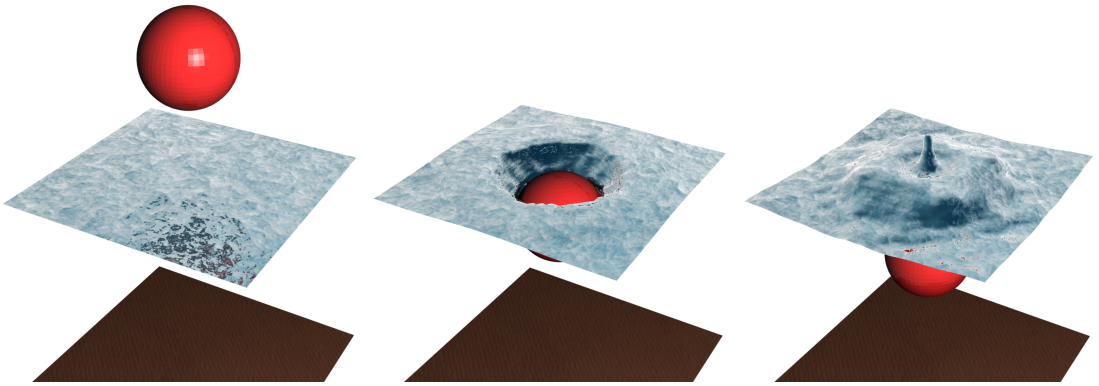
ρ_w	h_w	Δ	h_c^{sim}	h_w^{sim}	Δ^{sim}	ε_Δ	ε_h
500	4.2918	0.0	4.1684	4.1561	-0.0123	0.008	0.0316
600	4.3028	0.1006	4.0501	4.1320	0.0819	0.013	0.0397
700	4.3138	0.2051	3.9125	4.1291	0.2166	0.008	0.0428
800	4.3249	0.3193	3.8699	4.2021	0.3322	0.009	0.0284
900	4.3359	0.4563	3.7550	4.1771	0.4221	0.023	0.0366

(a) Numerical results for water height and sphere position

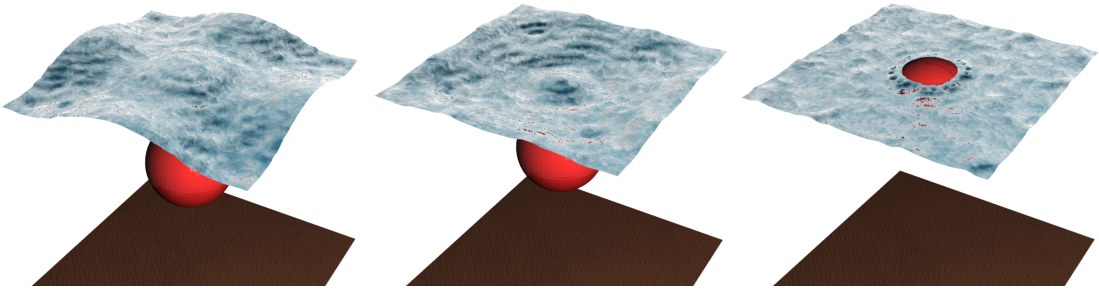


(b) Sketch

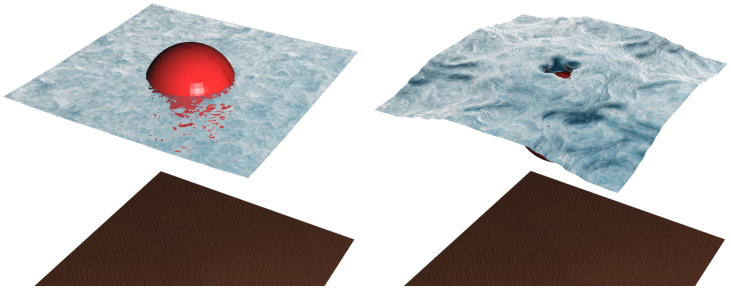
Figure 9.4: Analytical solution and numerical results for the static swimming position of the floating sphere



(a) $t = 0s$ (b) $t = 1s$ (c) $t = 2s$



(d) $t = 3s$ (e) $t = 4s$ (f) $t = 5s$



(g) $t = 6s$ (h) $t = 7s$

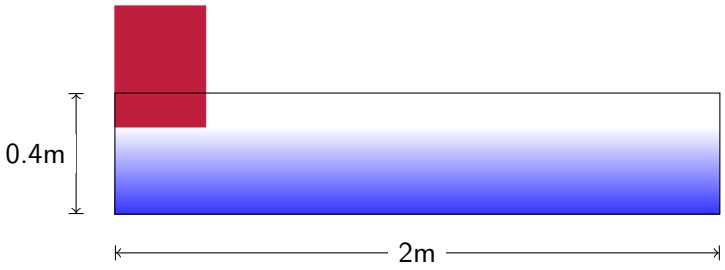
Figure 9.5: Time series for the immersing sphere and a sphere density of $\rho_s = 900\text{kg/m}^3$

9.4.2 Scott Russels wave generator

After the initial validations, we present a more complex scenario, where the time-evolution of the rigid-body-motion is known from experimental data. In 1844, Scott Russel was the first to discover the phenomenon of solitary waves in a shallow canal. He tried to reproduce his observations by scale experiments in a laboratory wave flume. Similar experiments have been carried out by Monaghan and Kos [119]. A rectangular solid block (weight 38.2 kg, 0.4m tall, 0.3m long, 0.39m wide) is falling in the water, creating a quasi-solitary wave. Monaghan and Kos [119] observed the vertical position of the block and the evolution of the water height at a probe location 1.2 meters away from the left end of the tank. Hence, opposite to the previous test case, not only the steady-state solution for the final block position (which apparently is on the bottom of the tank) but mainly the time-dependant block displacement can be validated. The model parameters are given in test case setup 9.2. We concentrate on simulating the flow along a slice of the wave tank in order to save computational costs.

Parameter	Value
Domain	2m x 0.4m
Water height	0.288m
Lattice	200 x 40
	400 x 80
Δx	0.01m, 0.05m
Ma	0.1, 0.01

(a) Parameters



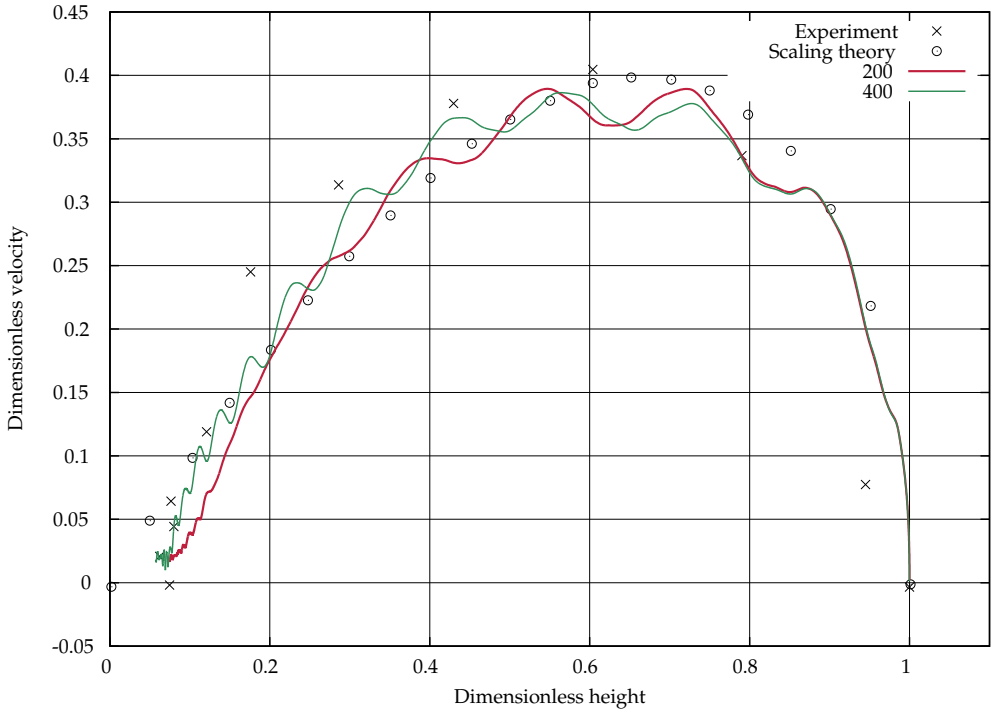
(b) Geometry

Test case 9.2: Scott Russels wave generator

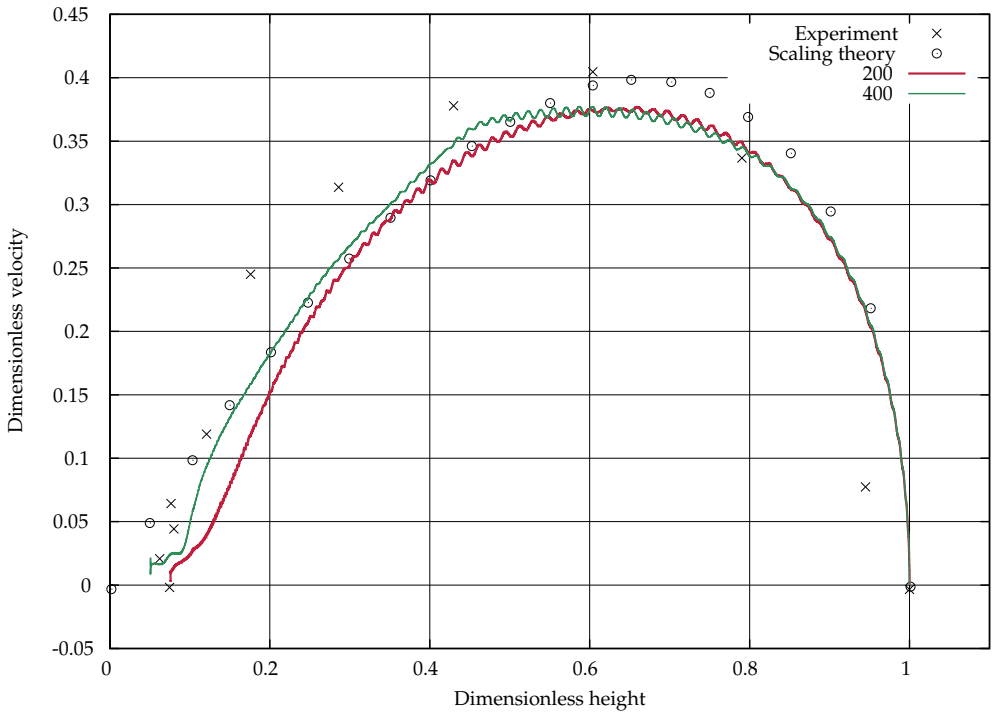
As the block begins to fall, a reverse plunging breaker at the block corner develops. In the following, the entrapped air package is advected by the flow, while the solitary wave starts to propagate. In our simulations, the air phase is not included but represented by pressure boundary conditions at the phase interface, so that - in the long run - the air package is expected to disappear (see also section 7.5.1).

Fig. 9.6 compares the numerical results for two different Mach numbers to the experimental data of Monaghan and Kos [119] and a scaling theory. In each plot, the block displacement for two different grid resolutions (20 and 40 lattice nodes per water height) are shown. For high Mach numbers of 0.1 the acoustic modes remarquably influence the simulation results and lead to an oscillatory behaviour. For lower Mach numbers, the oscillations are minimized and the overall block displacement reproduces the experimental data resp. the data obtained from scaling theory very well. The refinement in terms of grid spacing does not drastically improve the quality of the result anymore, so that convergence can be assumed.

In Fig. 9.7 and Fig. 9.8, snapshots of the simulation are given for several dimensionless points in time t', with $t' = t \cdot \sqrt{g/h}$ on the basis of the water depth h and gravity g.

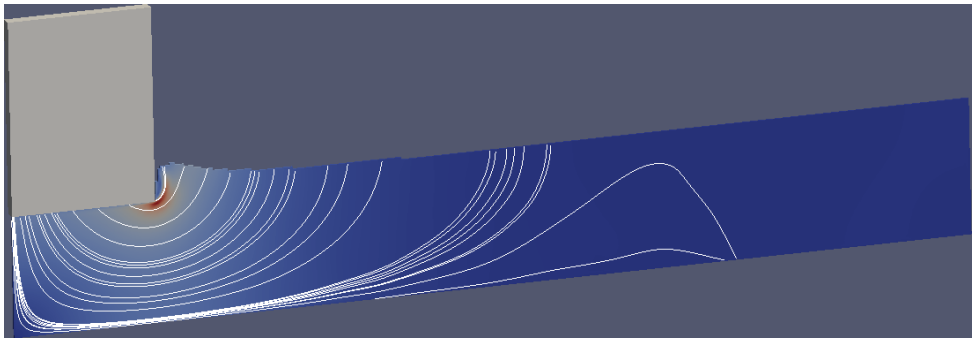


(a) Mach number 0.1

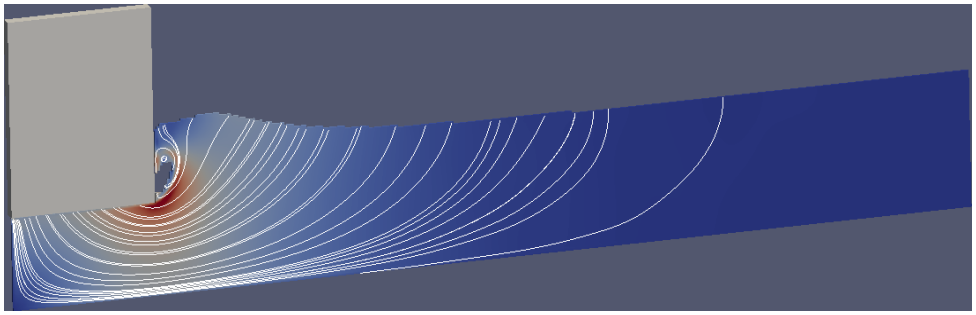


(b) Mach number 0.01

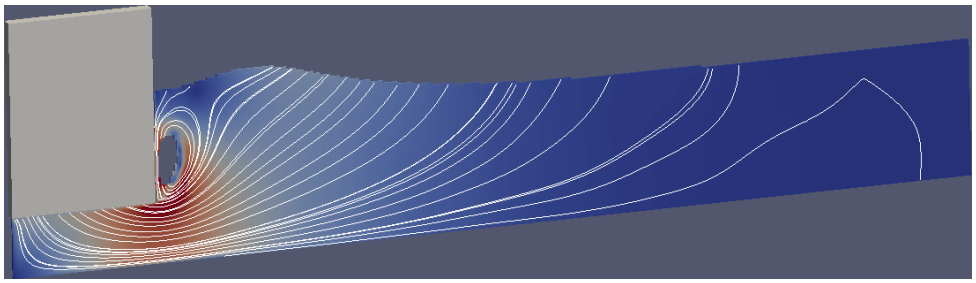
Figure 9.6: Scott Russel wave generator, comparison of numerical results and experimental data for the block displacement



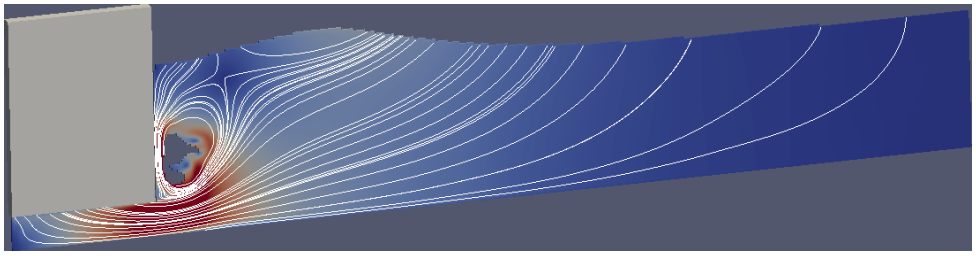
(a) $t' = 0.6$



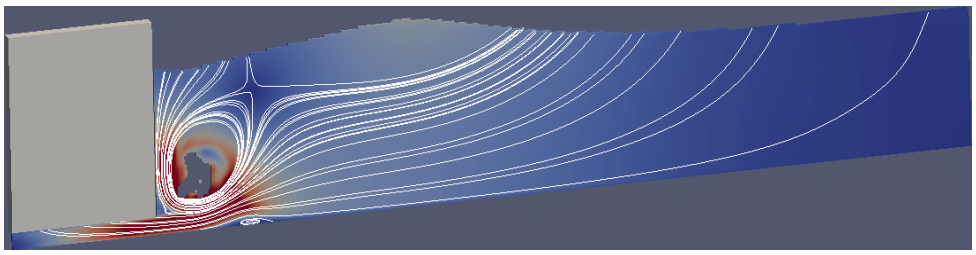
(b) $t' = 1.2$



(c) $t' = 1.8$

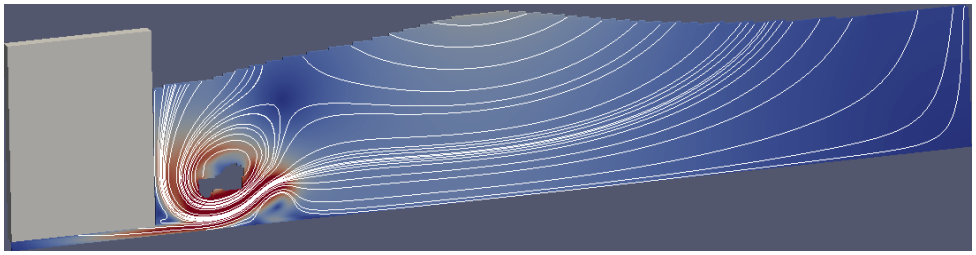


(d) $t' = 2.4$

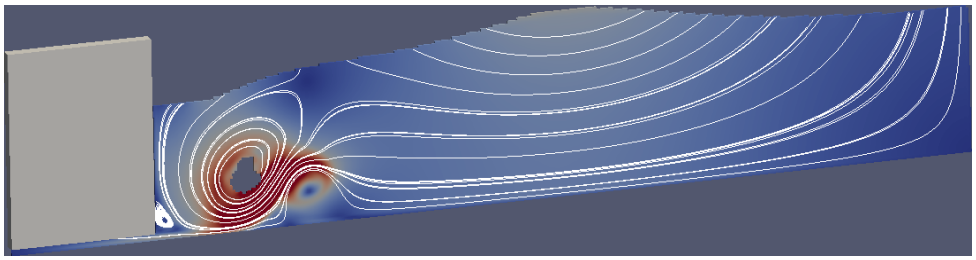


(e) $t' = 3.0$

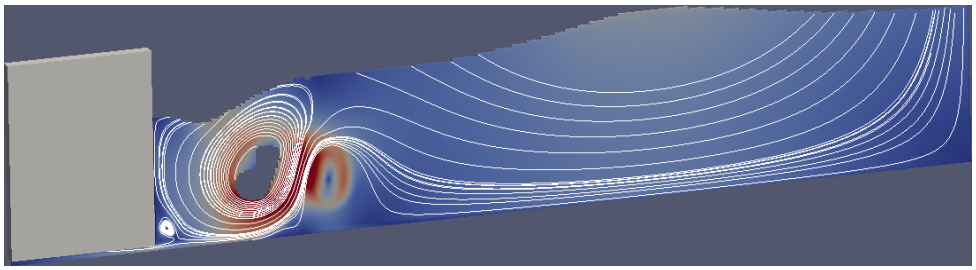
Figure 9.7: Scott Russel wave generator



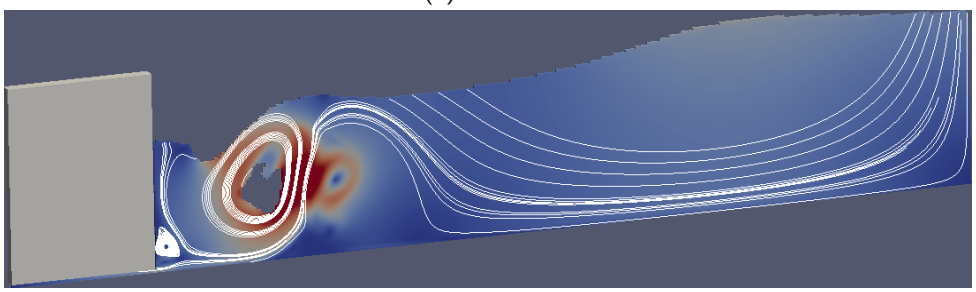
(a) $t' = 3.6$



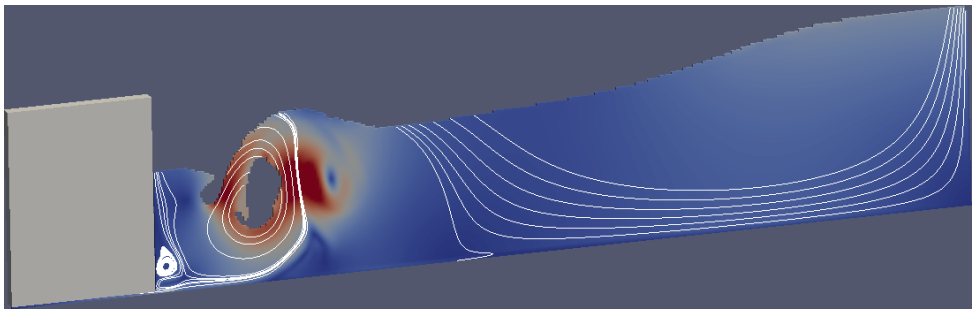
(b) $t' = 4.2$



(c) $t' = 4.8$



(d) $t' = 5.4$



(e) $t' = 6.0$

Figure 9.8: Scott Russel wave generator (Ctd.)

9.4.3 Wave impact on a box stack

As one possible application, test case setup 9.3 deals with the impact of a wave on a stack of boxes. The wave loading is created by a classical breaking dam scenario, in which a column of water in the rear part of the domain is allowed to collapse. The box stack consists of 15 boxes, which are modeled as rigid bodies in the PE physics engine, and are placed in the front part of the wave tank. The simulations were run for an artificial test case setup at Reynolds number 50000 and Froude number 1.5.

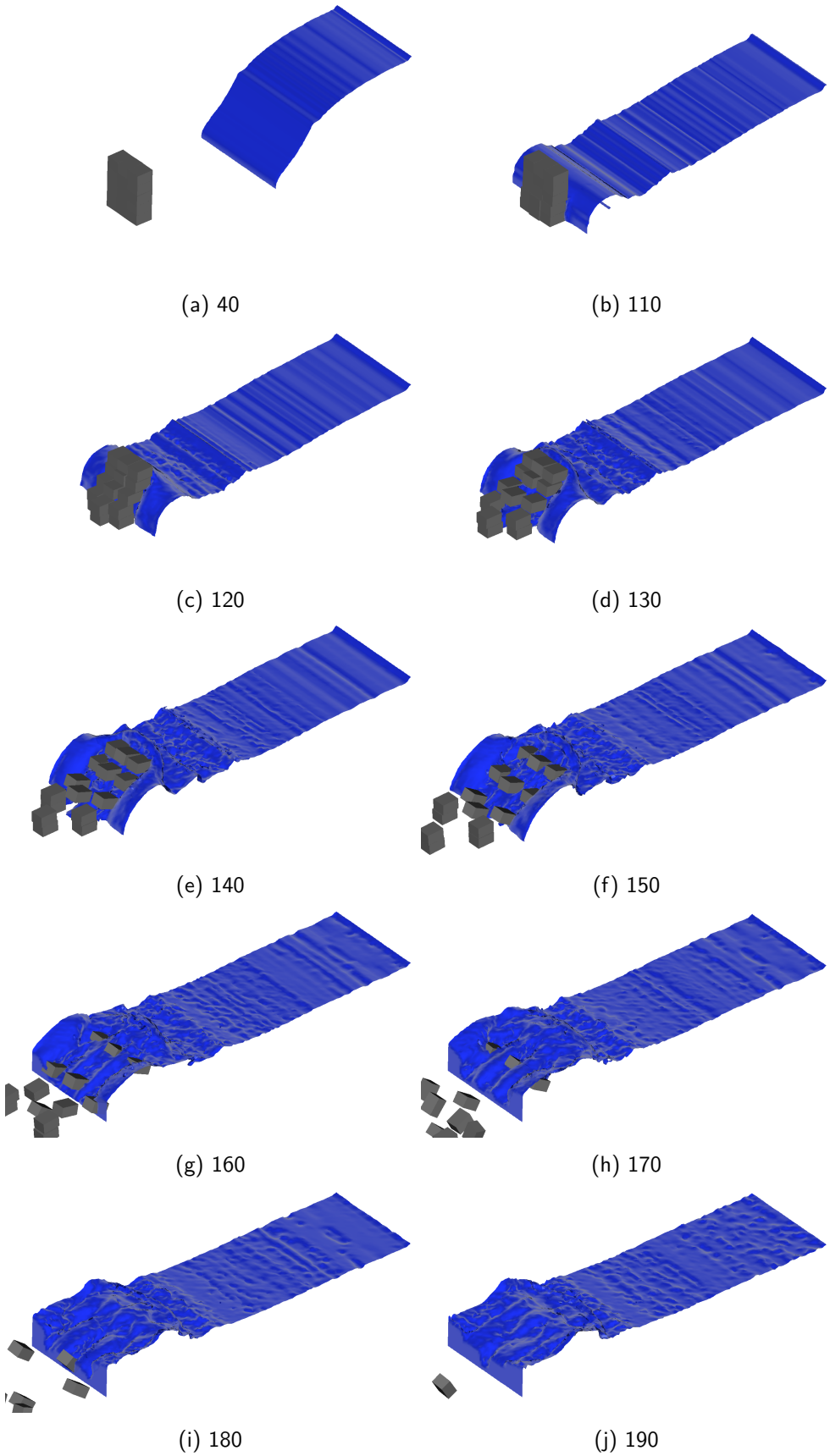
Parameter	Value
Lattice	$160 \times 48 \times 48$
Re	50000
Fr	1.5

(a) Parameters

(b) Geometry

Test case 9.3: Wave impact on a box stack

Even at low grid resolutions ($160 \times 48 \times 48$), realistic behavior can be observed. In Fig. 9.9 selected snapshots of the simulation are shown. As the surge front impacts on the box stack, the boxes are pushed forward and advected with the flow. This show case demonstrates the general applicability of the coupled approach to large-scale FSI-free surface problems involving more than one or two geometrical objects.



(a) 40 (b) 110

(c) 120 (d) 130

(e) 140 (f) 150

(g) 160 (h) 170

(i) 180 (j) 190

Figure 9.9: Wave impact on box stack for selected time steps

CHAPTER 10

Object-oriented implementation in VirtualFluids

The efficient numerical analysis of large-scale physical problems is very demanding, and in order to provide a flexible and expandable numerical tool, a sophisticated software concept is crucial. Apart from the short-term usability, the sustainability is the key point of intelligent software development. In this chapter, the integration of the previously described free surface capturing algorithms in the non-uniform, adaptive and massively parallel LB solver VirtualFluids is described.

10.1 The LB solver VirtualFluids

The current work extends the existing code-framework VirtualFluids and uses it as an environment for the development of the free surface algorithms. VirtualFluids is a Lattice Boltzmann based flow solver on the basis of block- structured, hierarchical, nonuniform grids. It has mainly been developed and validated at the iRMB by Freudiger [42] and Geller [49].

The solver has been used in numerous publications and has been extensively validated. Freudiger [42] shows basic validations and describes the massively parallel application of VirtualFluids and analyzes scalability. The extensibility of VirtualFluids to multiphase flow problems is the key point of his work. The massively-parallel simulation of rising bubbles on adaptively refined grids shows the high potential of the solver. In the field of fluid-structure interation, Geller [49] presents an FSI-coupling approach and its implementation in VirtualFluids. Finally, he presents three-dimensional fluid-structure interaction examples in a coupling to a high-order p-FEM-solver for complex, turbulent FSI benchmark problems. Uphoff [164] uses the code framework for the developent and analysis of turbulence models. Moreover, various student research projects have contributed to VirtualFluids, e.g. [88, 142, 26].

The main goal during the development of VirtualFluids was to combine high computational performance and parallel efficiency with basic principles of state-of-the-art software development, to guarantee sustainable usage and enhancement of the code and to decouple its life cycle from single

PhD theses and the ending of individual employments. This way, large amounts of the fundamental implementation work do not have to be repeated and a collective code basis can be used. Moreover, the extensive and distributed benchmarking leads to mutual benefits in terms of diversified bug searches and benchmarking.

VirtualFluids provides the basic routines for collision, forcing and propagation in a LB bulk scheme (chapter 3). Boundary conditions for the flow field can be specified on the basis of geometric primitives. The same holds for grid refinement regions. Apart from the specific numerical kernels, VirtualFluids provides numerous useful third-party packages, efficient container classes, a set of geometrical objects and many more features. For massively-parallel simulations, Freudiger [42] developed a service-oriented framework on the basis of the VirtualFluids class layout. The same base structures and algorithms of the serial code can be used for parallel simulations, without extensive code modifications. The inter-process communication is based on a connector-transmitter concept.

10.2 Free surface extension

During the implementation of the free surface kernel, a flexible software layout which can deal with multiple free surface models was of great importance. The straightforward extension to new models is crucial, as in the long run, the existing framework might also be extended to higher-order surface reconstruction schemes, second-order flux calculation and hybrid tracking or capturing schemes.

The class structure of VirtualFluids is subdivided into three layers: a topology layer, a grid layer and a physics layer (Fig. 10.1). Extensions to more complex physics beyond the simple LB bulk schemes are implemented by extending the physics layer, while using the same base classes for topology and grid management. For the free surface extension of VirtualFluids, mainly the basic classes which contain the physical properties of the simulation (block grid), which control the simulation (calculator) and which contain the lattice notes (block descriptors) have to be extended. It is remarquable that most of the free surface models have a lot of basic functions in common. To consider this, VirtualFluids is extended by a fourth logical layer, the *free surface layer*. It contains the specific operations needed for the free surface models, whereas the similarities are implemented in the main class structure of the free surface package in the "physics layer". The base class `FS3DBlockDescriptor` provides the basic methods for all free surface simulations, as e.g.

- algorithms for the free surface triangulation for visualization purposes
- free surface boundary conditions
- calculation of face- or cell-centered macroscopic values
- initialization procedures for new fluid nodes
- algorithms for the distribution of excessive mass

The same concept is applied to the `FS3DBlockGrid` and the `FS3DCalculator` classes in the physics layer. They do not contain model-specific free-surface attributes or functions, but are general and can handle all kinds of free surface representations.

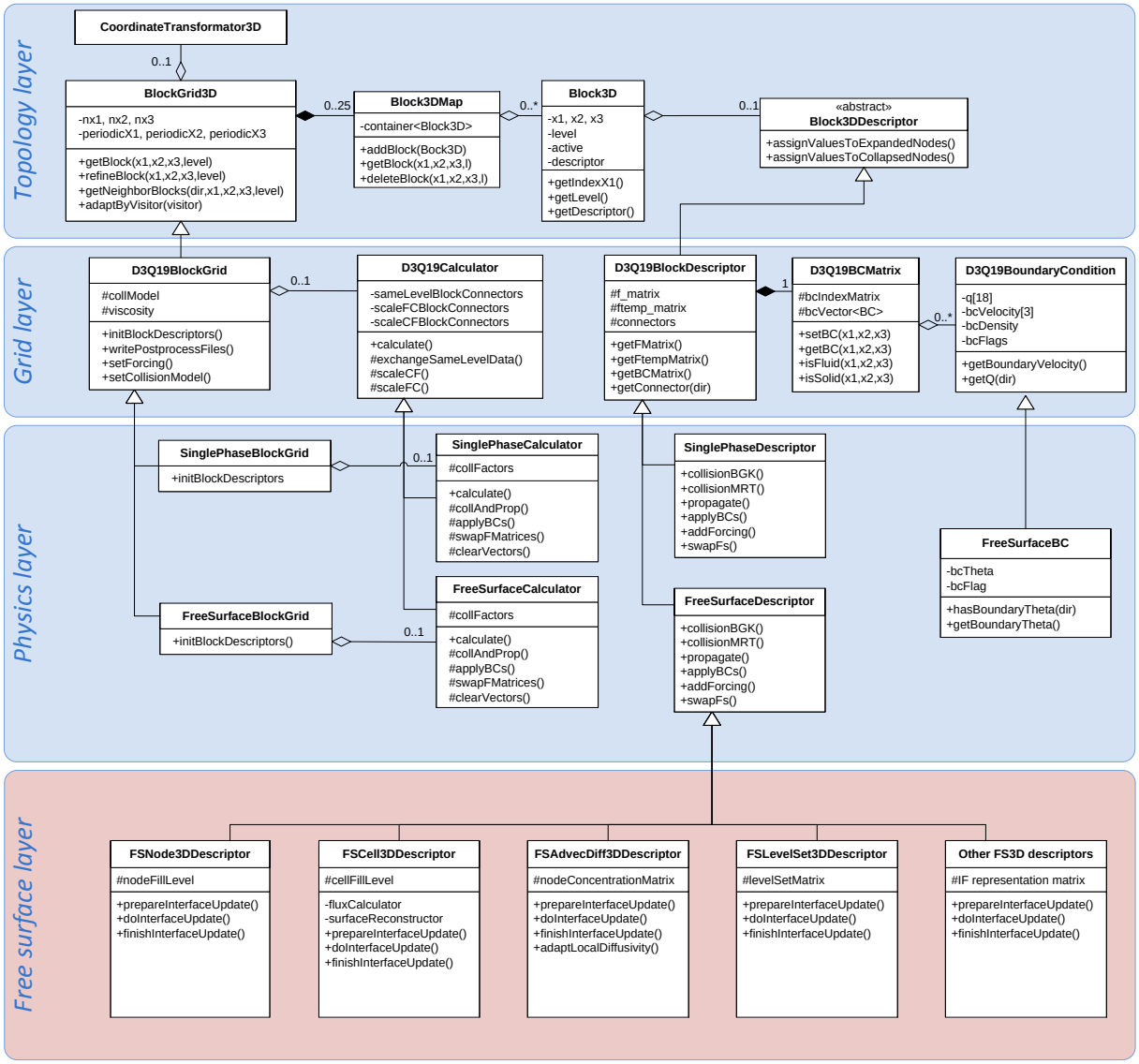


Figure 10.1: Four-layer class structure of VIRTUALFLUIDS *Free Surface Edition* (illustration on the basis of [42])

The specific features of each advection scheme are implemented in derived block descriptor classes. So far, special block descriptors and connector modules exist for four different free surface models:

- nodebased: implements the Lattice-Boltzmann free surface model on the basis of the work of Körner et al. [95]
- cellbased: implements the hybrid LBM-VOF-model
- levelset: provides basic Level-Set functionality
- advecdiff: uses a LBM advection-diffusion scheme for the advection of the phase interface

The block descriptors for the free surface models are extended by the main storage containers needed for the free surface representation, e.g. the fill level of a cell and normal vector information. The interface update itself is split up into three sub-steps (prepare, do and finish the interface update). These three parts are implemented as virtual functions in the abstract block descriptor base class, so that all free surface models have to implement them.

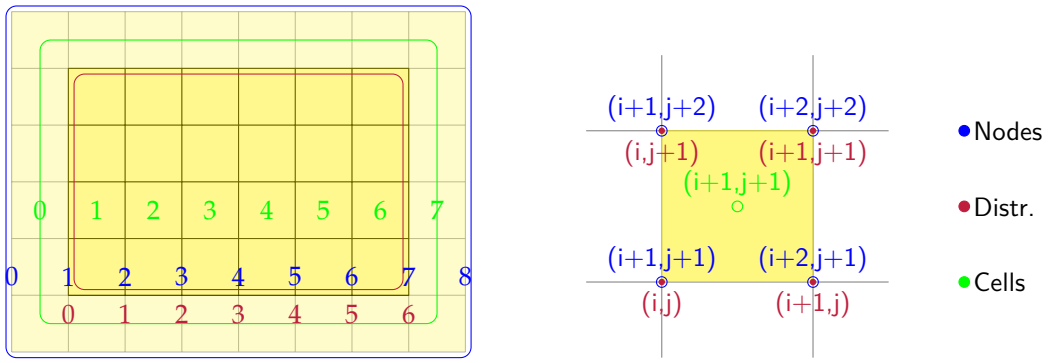
For the selection of a specific free surface model, traits are used [147]. A trait specifies the parts of the code which have to be used for the corresponding free surface model. The `FSCell3DTrait` for example selects the correct cellbased block descriptor type, in combination with a valid set of connector modules and initialization routines. This concept technically corresponds to a policy pattern, a strategy which is evaluated at compile time. The `FS3DTrait` is specified at compile time as a template argument for the `FS3DBlockGrid`, `FS3DCalculator` and also the `FS3DTestCase` in the physics layer. Hence, by changing one compiler option, the user can change between the free surface models and simulate identical test cases with different free surface models, using similar input parameter sets and model specifications. Confusions and misinterpretations due to the inconsistent choice of parameters are avoided.

10.3 Implementation details of the VOF extension

In this section, some details on the implementation of the new, hybrid LBM-VOF model will be given. The VOF-based block descriptor `FSCell3DBlockDescriptor` is extended by additional storage containers for cell properties (fill level, normal vector, ...) and node properties (maroscopic values, ...) on a staggered grid. For periodic boundary conditions and for the communication between blocks, ghost layers for the node- and cell-based free surface properties are used. Opposite to that, by definition, VIRTUALFLUIDS uses a parallelization approach for the particle distribution functions without ghost layers. This leads to the grid layout depicted in Fig. 10.2 and a quite complicated numbering with different running indeces for all three kinds of matrices. In this 2D cut, a VOF cell at position (i+1,j+1) consists of four nodes. The node states are accessible at memory location (i+2,j+1), (i+2,j+2), (i+1,j+2) and (i+1,j+1). To access the corresponding particle distribution functions, the index has to be decremented by one, i.e. (i+1,j), (i+1,j+1), (i,j+1) and (i,j).

The LBM calculations are done on all LB nodes. The update interface steps only are executed for the inner part of the domain. Cells in the ghost layer are not changed during the computation but only in the synchronization step.

A last detail on the implementation shall be mentioned: the VOF methodology is not directly attached to the corresponding block descriptor, but is sourced out to adapters and visitors, which are connected to the cellbased block descriptor. The calculation of the flux terms Φ_i is assigned to the



(a) Staggered grid (b) Staggered numbering

Figure 10.2: Ghost layers for state and fill level matrix

`FS3DFluxCalculator` class. Time-constant and time-linear flux calculators are available, whereas all classes have to implement functions for the flux calculation with or without an additional storage containers for the flux terms. During the interface update, a request for the flux calculation for specific interface cells is sent from the block descriptor to the flux calculator. The flux calculation itself is based on a proper surface reconstruction. As the latter is also needed in the visualization step, the surface reconstruction object is assigned to the block descriptors (`FS3DSurfaceReconstructor`). Constant and PLIC surface reconstructors are available. Each reconstructor has to implement two main geometric functions: the evaluation of the moistened face area, which is needed for time-constant flux calculation, and the intersection of the reconstructed surface with arbitrary cuboids, as needed for the higher-order flux calculation. Extensions to higher-order reconstruction schemes are straightforward, as long as these two interfaces are implemented. This way, the extension of the code to e.g. a spline-based surface reconstruction is straightforward: a proper surface reconstruction module has to be implemented and attached to the block descriptor, no more additional changes are needed.

Exemplarily, the resulting algorithm for the free surface scheme of chapter 7 is given in Alg. 10.1.

Algorithm 10.1: Update interface framework

```
foreach block descriptor do
    collision;
    forcing;
    propagation;
end
SYNCPOINT I: exchange particle distribution functions and new fill levels;
foreach block descriptor do
    apply boundary conditions;
    set node matrices for density and velocity;
    set cell normal vector;
end
SYNCPOINT II: exchange normal vector information and node velocities in the ghost
layer;
foreach block descriptor do
    set node matrices for density and velocity used in the node initialization step;
    calculate fluxes and evaluate the new fill level of the interface cells;
    set new cell states;
end
SYNCPOINT III: exchange new node and cell states and init-matrix;
foreach block descriptor do
    initialize new fluid nodes;
    distribute mass, if applicable;
end
```

CHAPTER 11

Conclusions and outlook

As announced in the introduction, the main intention of this work was to develop a set of numerical tools for the simulation of free surface flow problems. Different length and time scales had to be addressed, and the implementation of the algorithms was based on state-of-the-art high performance computing hardware. In general, apart from concrete results, in this thesis it is shown that models on the basis of the Lattice Boltzmann equation are suitable for the simulation of non-linear free surface flow. High performance implementations on clusters and GPUs have been presented. Moreover, the method has shown to be capable of handling complex geometries and large topological changes.

11.1 Summary

In the first part of this thesis, a Lattice Boltzmann model for the simulation of shallow water flows has been implemented and applied. Very good results were achieved and the increase of performance using GPU hardware was remarkable. The tsunami runup onto a plane beach demonstrated the capability of the implementation to handle wave propagation over distances of 10 to 50km and the subsequent runup. Secondly, the numerical simulation of the runup onto a complex three-dimensional beach roughly reproduced the data which was observed during scale experiments in an experimental wave tank.

Subsequently, the GPU implementation of a three-dimensional numerical wave tank for the simulation of turbulent flow was presented. The model was successfully validated with several benchmark problems and could be applied to typical flow problems in civil engineering, as e.g. the simulation of turbulent flow around a weir, wave impact simulations and wave runup studies. For the wave generation, a piston-type wavemaker was used, resulting in a unidirectionally coupled fluid-structure interaction problem involving topological changes. The resulting waves have shown to be sufficiently accurate to reproduce runup and wave impact events. The drawbacks of the method are compensated by its simplicity which allowed for the efficient implementation on GPU hardware and the use of high

grid resolutions. The simulation of a wave impact on the skyline of South Manhattan demonstrated the applicability of the GPU wave tank to huge domains and geometries.

The resulting overall performance of both solvers on GPU hardware was found to be at least one order of magnitude higher than of shared-memory-parallel CPU LB kernels on a modern multi-core architecture. The ratio of the numerical simulation runtimes and the real-world time scale of the actual problem could be reduced to a factor of ten for some of the 3D applications and even below unity for wave propagation problems using the reduced models on the basis of the shallow water equations.

The linchpin of the second part of this work was the development of an enhanced free surface capturing scheme on the basis of a hybrid LBM-VOF method. We proposed a Volume-Of-Fluid (VOF) interface capturing algorithm with a piecewise linear, geometric reconstruction of the free surface (PLIC). The surface normal is obtained from the discrete fill level information using a finite-difference approximation on the basis of Parker-Youngs weighting factors with a spatial accuracy of $\mathcal{O}(\Delta x)$ to $\mathcal{O}(\Delta x^2)$. For the time-discretization, an explicit Euler scheme is used, whereas the time-dependent change of the free surface position during the advection is considered in the geometrical flux calculation scheme. The resulting interface capturing algorithm has successfully been coupled to VirtualFluids, a three-dimensional Lattice Boltzmann solver on the basis of non-uniform, block-structured grids. The LB solver provides the solution of the flow field in terms of velocity and density fields, while the VOF scheme updates the interface location. The new free surface capturing approach has shown to cope with various kinds of free surface flow problems, including the very demanding test case of a breaking wave. The resulting hybrid model has been validated with various benchmark test cases and experimental results, showing its validity and applicability.

Apart from the model development and validation itself, the coupling of the hybrid LB-VOF solver to a boundary element solver on the basis of potential flow theory (numerical wave tank, NWT) has been discussed, implemented and validated. In a weak coupling, the LB domain is initialized with results from the NWT to generate realistic and complex wave profiles. This way, even a breaking wave during shoaling on a slope, which is a demanding test case for VOF solvers due to high interface curvature and velocity gradients, was successfully simulated. Oppositely, the strong coupling is based on a perturbation approach, in which the velocity and pressure fields are split up into an inviscid part and a viscous perturbation. The inviscid flow solution is provided by linear wave theory, whereas the LB solves for the perturbation fields only. The coupling has been realized by a modified LB collision operator and was validated for the Eulerian drift in a wave-induced boundary layer under progressive waves.

Finally, a bidirectional, partitioned coupling to a rigid body engine for the simulation of FSI problems has been established. In this explicit coupling, the forces acting on the obstacles in the flow are sent to a structural solver. The structural response is calculated and influences the flow field via geometry changes and modified boundary conditions. A sphere, which is falling into a basin of water, was successfully simulated, and the capability of the method to handle large topological changes was demonstrated. Ultimately, the model was validated with the transient reference data from experiments of the Russel wave generator.

11.2 Future work

The implementation of two Lattice Boltzmann schemes on GPGPU hardware offers numerous possibilities for future work, both for the shallow water model and for the full three-dimensional flow.

For more realistic simulations, a more complete shallow water model including the effects of bottom- and top-shear stress, Coriolis force and turbulence has to be implemented. Moreover, the wave runup on complex beaches revealed several drawbacks of the current model and of the implementation, so that higher-order runup models have to be tested. Finally, a multi-GPU implementation should be addressed. LB simulations on multiple GPUs in a massively parallel cluster already have successfully been run at the iRMB. In the long term, recent developments in the field of non-uniform grid refinement should also be transferred to the shallow water GPU implementations. Recently, a patch-based, very flexible grid refinement technique has been proposed by Geier et al. [48]. This approach has been used and validated for non-uniform simulations of the flow around a cylinder, on the basis of a D2Q9 Lattice Boltzmann model [145]. Hence the idea of adapting it for the LB shallow water model is tempting.

Concerning the three-dimensional GPU free surface solver, more sophisticated wave generation methods have to be implemented. So far, the confined accuracy of Goring's solution for the wave maker limits the benchmark results. Additionally, bidirectional fluid-structure interaction phenomena in a GPU framework have to be addressed in detail. Initial work has been done, as the wave generation by means of a piston wave maker is a fluid-structure interaction (FSI) problem, and also the algorithms for the on-the-fly evaluation of forces on obstacles have been implemented. Nevertheless, the interaction of free surface flow with more complex moving geometries as e.g. ship propellers is still challenging. Similar to the shallow water GPU kernel, the parallelization of the three-dimensional kernel would also benefit from the huge computational power which is available in recent GPU clusters. With a multi-GPU free surface implementation, problems as the simulation of wave impact on South Manhattan could be addressed with a grid spacing below 50cm on uniform grids, or even below that, if the parallelization is realized in combination with non-uniform grid refinement.

The second part of this work was dealing with the development of a hybrid LBM-VOF free surface scheme. Concerning the advection scheme itself, extension to higher order surface reconstruction schemes, second-order flux calculation and hybrid tracking or capturing schemes could further improve the free surface representation. Especially the consideration of surface tension, which is based on a proper calculation of the interface curvature, is demanding. Local height functions, least-squares approaches or hybrid VOF-Level-Set methods seem to be most promising to estimate the curvature of the free surface. In this context, once the surface tension terms are added, the free surface model might be extended to deal with a second fluid phase. In modern multiphase methods on the basis of diffusive interfaces, the high density ratio between the air and the water phase still is challenging. A method on the basis of a sharp interface representation and subdomains which are coupled via boundary conditions at the phase interface might lead to more stable schemes. This way, up-to-date multiphase-flow-related problems in the field of air-sea interaction could be addressed. These types of problems would also benefit from the present coupling to the numerical wave tank.

This coupling to a solver on the basis of potential flow theory opens up uncountable new applications. So far, the weak coupling has served to further validate the advection scheme with the test case of a

breaking wave during shoaling. This methodology can easily be extended to three dimensions, as the underlying free surface capturing scheme and the LB framework are three-dimensional, anyway, and only were confined by periodic boundary conditions in the third space direction. Hence the generation of three-dimensional freak waves, for example, for wave impact studies on slender structures can be examined. The strongly coupled simulations with a modified MRT collision operator also has shown great promise for further extensions and applications. Here, the implementation on GPUs in order to simulate the Eulerian drift in real world wave propagation problems would be the next step. Moreover, the coupling to the NWT demonstrated that, apart from the classical adaption of the grid spacing, hybrid models in which the complexity of the flow solvers is adapted to the particular requirements in each subdomain is a key point in the development of future methods. Hybrid models on the basis of coupled shallow-water and full three-dimensional models should also be addressed.

In the field of fluid-structure interaction, the initial free surface validations are only the starting point of various activities in the hybrid FSI-free surface flow area. Recently, Geller et al. examined the erosion effects in river beds created by a ship propeller during the startup phase. This test case would definitely benefit from a free surface modeling, instead of impermeable, non-movable boundary conditions at the upper domain boundary. Secondly, the coupling of VirtualFluids to the higher-order pFEM-solver AdHoC can be enriched with free surface functionality, as already has been shown by Kollmannsberger, with the simulation of wave impact on a flexible and slender tower structure. All these applications do not demand complex extensions of the currently available coupling scheme, but only minor modifications in terms of geometrical operations. Concerning the VOF free surface capturing scheme in the FSI context, the current treatment of the triple-point at the air-water-solid-interface is poor. More detailed interface reconstruction is needed. The theory is widely available, although the universal implementation for arbitrary 3D geometries is challenging. Nevertheless, the general validity of the explicit bidirectional coupling including a free surface was shown.

APPENDIX A

Transformation matrix (MRT)

$$
\begin{bmatrix}
1\cdot & (1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1) \\
c^2\cdot & (-1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1) \\
c^4\cdot & (1 & -2 & -2 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1) \\
c\cdot & (0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0) \\
c^3\cdot & (0 & -2 & 2 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0) \\
c\cdot & (0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1) \\
c^3\cdot & (0 & 0 & 0 & -2 & 2 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1) \\
c\cdot & (0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1) \\
c^3\cdot & (0 & 0 & 0 & 0 & 0 & -2 & 2 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1) \\
c^2\cdot & (0 & 2 & 2 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -2 & -2 & -2 & -2) \\
c^4\cdot & (0 & -2 & -2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -2 & -2 & -2 & -2) \\
c^2\cdot & (0 & 0 & 0 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0) \\
c^4\cdot & (0 & 0 & 0 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0) \\
c^2\cdot & (0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0) \\
c^2\cdot & (0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1) \\
c^2\cdot & (0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0) \\
c^3\cdot & (0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0) \\
c^3\cdot & (0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1) \\
c^3\cdot & (0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1)
\end{bmatrix}
$$

APPENDIX B

Piecewise linear interface reconstruction (PLIC)

A piecewise linear interface reconstruction method (PLIC) represents the surface as a line segment (2D) or a plane (3D), which uniquely can be described by its normal vector and the fill level of a cell. For the sake of simplicity, this method is first presented and explained in 2D before extending it to three dimensions. For all the computations, the normal vector has to be mirrored and sorted, so that finally $n_1 \leq n_2 \leq n_3$ and $n_i \geq 0.0$ hold. The permutation has to be stored to be able to assign the correct moistened face values after the surface reconstruction.

B.1 Introduction in two dimensions

In 2D the interface in each cell is characterized by a line segment according to

$$\mathbf{n} \cdot \mathbf{x} = n_x \cdot x + n_y \cdot y = \alpha \tag{B.1}$$

If the normal vector has unit length, α corresponds to the distance of the free surface to the origin of the considered cell. In the volume of fluid approach, the cell fill level ε is known and the normal direction of the surface $\mathbf{n}$ can be approximated by e.g. using finite differences, as elaborately discussed in section 6.2.1. In the following step, the unknown parameter α has to be calculated. The derivation is based on the assumption $n_1 \leq n_2$ in order to avoid additional case distinctions. In the implementation, this can be achieved by a simple permutation. For the axis intercepts, the following holds:

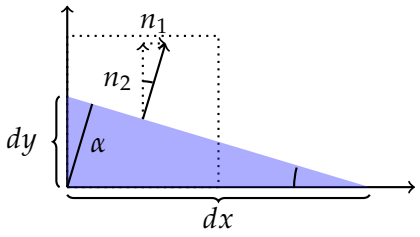


Figure B.1: PLIC

$$\sin(\beta) = \frac{n_1}{|\mathbf{n}|} = \frac{\alpha}{dx} \tag{B.2}$$

$$\cos(\beta) = \frac{n_2}{|\mathbf{n}|} = \frac{\alpha}{dy} \tag{B.3}$$

$$\Rightarrow dx = \frac{\alpha \cdot |\mathbf{n}|}{n_1} = \frac{\alpha}{n_1} \tag{B.4}$$

$$\Rightarrow dy = \frac{\alpha \cdot |\mathbf{n}|}{n_2} = \frac{\alpha}{n_2} \tag{B.5}$$

After this preliminary work, the cut area A below the line segment and the unit cell as the 2D equivalent to a cell fill level is computed, before inverting the resulting expression in order to determine α. The area of the basic fluid triangle is

$$A_0 = \frac{1}{2} \cdot \frac{\alpha}{n_1} \cdot \frac{\alpha}{n_2} = \frac{1}{2} \cdot dx \cdot dy \tag{B.6}$$

When the fill level increases and the base lines are completely moistened, the triangle area outside the control volume (red area in Fig. B.2) has to be subtracted.

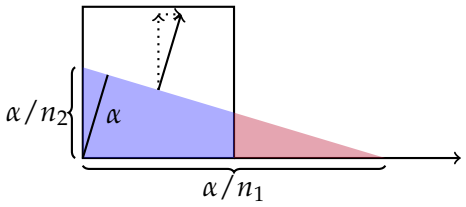


Figure B.2: Different area fractions

This phenomenon occurs once for each direction. The area fractions which have to be subtracted read

$$A_1 = 0.5 \cdot \left(\frac{\alpha}{n_1} - \Delta x_1\right) \cdot \left(\frac{\alpha/n_1 - \Delta x_1}{\alpha/n_1} \cdot \frac{\alpha}{n_2}\right) \tag{B.7}$$

$$= 0.5 \cdot \left(\frac{\alpha}{n_1} - \Delta x_1\right)^2 \cdot \frac{n_1}{n_2} \tag{B.8}$$

$$= 0.5 \cdot \frac{(\alpha - n_1 \Delta x_1)^2}{n_1 n_2} \tag{B.9}$$

$$A_2 = 0.5 \cdot \frac{(\alpha - n_2 \Delta x_2)^2}{n_1 n_2} \tag{B.10}$$

Mathematically, the following holds:

$$A(\alpha, \mathbf{n}) = \frac{\alpha^2}{2 n_1 n_2} \left[1 - H(\alpha - n_1 \Delta x_1) \left(\frac{\alpha - n_1 \Delta x_1}{\alpha}\right)^2 - H(\alpha - n_2 \Delta x_2) \left(\frac{\alpha - n_2 \Delta x_2}{\alpha}\right)^2\right], \tag{B.11}$$

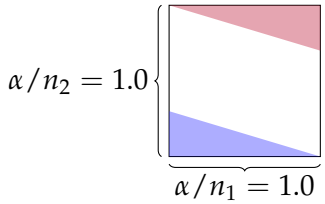
with Heaviside function H, which expresses the two discontinuities, where the corner points of the control volume are moistened and a part of the triangle area is outside the control volume:

$$H(x) = \begin{cases} 1, & \text{for } x > 0 \\ 0, & \text{for } x < 0 \end{cases} \tag{B.12}$$

If the VOF control volumes are assigned to the unit cells of the LB lattice, leading to a cell spacing of $\Delta x_i = 1$, the expression for A reduces to

$$A(\alpha, \mathbf{n}) = \frac{\alpha^2}{2n_1n_2}\left[1 - H(\alpha - n_1)\left(\frac{\alpha - n_1}{\alpha}\right)^2 - H(\alpha - n_2)\left(\frac{\alpha - n_2}{\alpha}\right)^2\right] \tag{B.13}$$

In a standard VOF approach, the normal vector and the fill level are known, and α is the only unknown parameter. Hence, in order to compute α, Eq. B.13 now has to be inverted. Due to the two Heaviside functions, the functional is not continuously differentiable and problems in the inversion occur. A case distinction is needed to determine α in the regions between the two *critical areas*. These critical areas correspond to the two states, where the two Heaviside functions switch from zero to one. Inserting $\alpha = n_i$ into Eq. B.13 leads to the resulting critical areas given in Eq. B.14 and Eq. B.15.



$$A_1 = \frac{\alpha^2}{2n_1n_2} = \frac{n_1}{2n_2} \tag{B.14}$$

$$A_2 = \frac{\alpha^2}{2n_1n_2}\left(1 - \frac{\alpha - n_1}{\alpha}\right) = \frac{2n_2 - n_1}{2n_2} \tag{B.15}$$

The unknown line parameter α can be obtained via a piecewise inversion of Eq. B.13, for three different cases:

First case: $A < A_1$, both Heaviside functions are zero ($\alpha < n_1$ and $\alpha < n_2$):

$$A = \varepsilon = \frac{\alpha^2}{2n_1n_2}\left[1 - H(\alpha - n_1)\left(\frac{\alpha - n_1}{\alpha}\right) - H(\alpha - n_2)\left(\frac{\alpha - n_2}{\alpha}\right)\right] \tag{B.16}$$

$$= \frac{\alpha^2}{2n_1n_2} \tag{B.17}$$

$$\Rightarrow \alpha = \sqrt{2An_1n_2} = \sqrt{2\epsilon n_1n_2} \tag{B.18}$$

Second case: $A_1 < A < A_2$, one Heaviside function is one ($\alpha > n_1$ and $\alpha < n_2$):

$$A = \varepsilon = \frac{\alpha^2}{2n_1n_2}\left[1 - H(\alpha - n_1)\left(\frac{\alpha - n_1}{\alpha}\right) - H(\alpha - n_2)\left(\frac{\alpha - n_2}{\alpha}\right)\right] \tag{B.19}$$

$$= \frac{\alpha^2}{2n_1n_2}\left[1 - \left(\frac{\alpha - n_1}{\alpha}\right)^2\right] \tag{B.20}$$

$$= \frac{\alpha^2 - (\alpha - n_1)^2}{2n_1n_2} = \frac{\alpha^2 - \alpha^2 + 2\alpha n_1 - n_1^2}{2n_1n_2} = \frac{2\alpha n_1 - n_1^2}{2n_1n_2} \tag{B.21}$$

$$\Rightarrow \alpha = \frac{2n_1n_2\epsilon - n_1^2}{2n_1n_2} = \frac{2n_2\epsilon - n_1}{2n_1n_2} \tag{B.22}$$

Third case: $A > A_2$, both Heaviside functions are one ($\alpha > n_1$ and $\alpha > n_2$):

$$\begin{aligned} A &= \varepsilon = \frac{\alpha^2}{2n_1n_2}\left[1 - H(\alpha - n_1)\left(\frac{\alpha - n_1}{\alpha}\right)^2 - H(\alpha - n_2)\left(\frac{\alpha - n_2}{\alpha}\right)^2\right] && \text{(B.23)}\\ &= \frac{\alpha^2}{2n_1n_2}\left[1 - \left(\frac{\alpha - n_1}{\alpha}\right)^2 - \left(\frac{\alpha - n_2}{\alpha}\right)^2\right] && \text{(B.24)}\\ &= \frac{\alpha^2 - (\alpha - n_1)^2 - (\alpha - n_2)^2}{2m_1m_2} && \text{(B.25)}\\ &= \frac{\alpha^2 - \alpha^2 + 2\alpha n_1 - n_1^2 - \alpha^2 + 2\alpha n_2 - n_2^2}{2m_1m_2} && \text{(B.26)}\\ \Leftrightarrow 2n_1n_2A &= 2\alpha(n_1 + n_2) - (n_1^2 + n_2^2) - \alpha^2 && \text{(B.27)}\\ 0 &= \alpha^2 - 2\alpha(n_1 + n_2) + (n_1^2 + n_2^2) + 2n_1n_2A && \text{(B.28)}\\ & && \text{(B.29)} \end{aligned}$$

Solving this quadratic equation with p-q-formula leads to the following expression for α:

$$\begin{aligned} \alpha &= (n_1 + n_2) \pm \sqrt{(n_1 + n_2)^2 - ((n_1^2 + n_2^2) + 2n_1n_2A)} && \text{(B.30)}\\ &= (n_1 + n_2) \pm \sqrt{n_1^2 + 2n_1n_2 + n_2^2 - n_1^2 - n_2^2 - 2n_1n_2A)} && \text{(B.31)}\\ &= (n_1 + n_2) \pm \sqrt{2n_1n_2 - 2n_1n_2A)} && \text{(B.32)}\\ &= (n_1 + n_2) \pm \sqrt{2n_1n_2(1 - \epsilon))} && \text{(B.33)} \end{aligned}$$

These three explicit expressions for α are simple to implement; only a three-case-distinction is needed. They are also used in the three-dimensional PLIC, in case of a vanishing third component of the normal vector.

B.2 Extension to three dimensions

The derivation for the general, three-dimensional configuration is more challenging, as six Heaviside functions appear. However, at least the derivation of the length dx, dy and dz may be adapted from the two-dimensional case and leads to the geometry in Fig. B.3.

In analogy to Eq. B.13, the expression for the volume below a plane yields:

$$V = \frac{1}{2n_1n_2n_3}\left[\alpha^3 - \sum_{j=1}^{3} H(\alpha - n_j\Delta x_j)\left(\alpha - n_j\Delta x_j\right)^3 + \sum_{j=1}^{3} H(\alpha - \alpha_{max} + n_j\Delta x_j)\left(\alpha - \alpha_{max} + n_j\Delta x_j\right)^3\right] \quad \text{(B.34)}$$

with $\alpha_{max} = \sum_{j=1}^{3} n_j\Delta x_j$, and for a LBM unit cell with $\Delta x_j = 1.0$ can be simplified to

$$V = \frac{1}{2n_1n_2n_3}\left[\alpha^3 - \sum_{j=1}^{3} H(\alpha - n_j)\left(\alpha - n_j\right)^3 + \sum_{j=1}^{3} H(\alpha - \alpha_{max} + n_j)\left(\alpha - \alpha_{max} + n_j\right)^3\right] \quad \text{(B.35)}$$

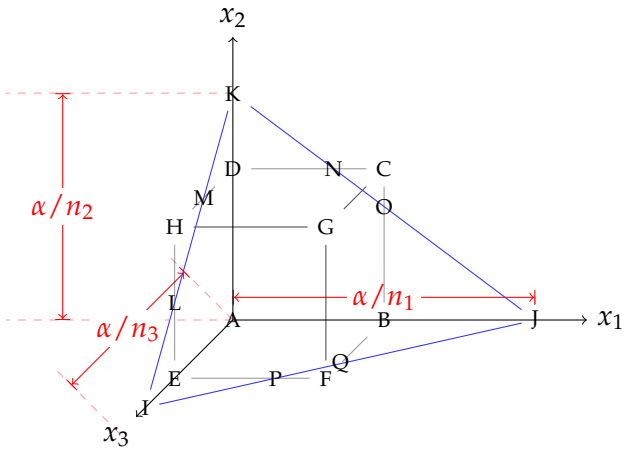


Figure B.3: Plane cutting a three-dimensional cube

with $\alpha_{max} = \sum_{j=1}^{3} n_j$. Thanks to the six Heaviside functions in Eq. B.35, six critical *volumes* have to be determined. In 2D the order of switching was clear due to the assumption of $n_1 \leq n_2$. In three dimensions, an additional case distinction is mandatory, see the Heaviside terms in Tab. B.1. Due to the assumption $n_1 \leq n_2 \leq n_3$ at least five of the six Heaviside terms may be sorted: $n_1 \leq n_2 \leq n_3 \leq n_1 + n_3 \leq n_2 + n_3$. For the sixth term a case distinction depending on the value of $n_1 + n_2$ is necessary.

Term	$j = 1$	$j = 2$	$j = 3$
$H(\alpha - n_j)$	$\alpha - n_1$	$\alpha - n_2$	$\alpha - n_3$
$H(\alpha - \alpha_{max} + n_j)$	$H(\alpha - n_2 - n_3)$	$H(\alpha - n_1 - n_3)$	$H(\alpha - n_1 - n_2)$

Table B.1: Six Heaviside functions in three dimensions

After some algebra, the six critical volumes can be calculated as:

$$V_1 = \frac{1}{6}\frac{n_1^2}{n_2 n_3} \tag{B.36}$$

$$V_2 = \frac{1}{6}\frac{n_1^2 + 3n_2^2 - 3n_1 n_2}{n_2 n_3} \tag{B.37}$$

$$V_3 = \begin{cases} \frac{1}{2}\frac{n_1+n_2}{n_3} & n_1 + n_2 < n_3 \\ \frac{1}{6}\frac{n_2^3+n_1^3-n_3^3+3n_2n_3^2+3n_1n_3^2-3n_2^2n_3-3n_1^2n_3}{n_1 n_2 n_3} & n_1 + n_2 > n_3 \end{cases} \tag{B.38}$$

$$V_4 = 1 - V_3 \tag{B.39}$$

$$V_5 = 1 - V_2 \tag{B.40}$$

$$V_6 = 1 - V_1 \tag{B.41}$$

The resulting values for α now can be calculated and are given in Tab. B.2.

Region	Result
$V < V_1$	$\alpha = \sqrt[3]{6n_1n_2n_3\varepsilon}$
$V_1 < V < V_2$	$\alpha = \frac{n_1}{2} - \frac{1}{6}\sqrt{-3n_1n_2 + 72n_2n_3\varepsilon}$
$V_2 < V < V_3$	α calculated with iterative algorithm
$V_3 < V < V_4$	$\alpha = \frac{1}{2}n_1 + \frac{1}{2}n_2n_3\varepsilon$, if $n_1 + n_2 < n_3$
	α calculated with iterative algorithm, if $n_1 + n_2 > n_3$
$V_4 < V < V_5$	α calculated with iterative algorithm
$V_5 < V < V_6$	$\alpha = \alpha_{max} - (\frac{n_1}{2} - \frac{1}{6}\sqrt{-3n_1n_2 + 72n_2n_3(1-\varepsilon)})$
$V > V_6$	$\alpha = \alpha_{max} - \sqrt[3]{6n_1n_2n_3(1-\varepsilon)}$

Table B.2: Analytical solutions for the plane coefficient α, 3D case

Bibliography

[1] Stéphane Abadie, Denis Morichon, Stéphan Grilli, and Stéphane Glockner. Numerical simulation of waves generated by landslides using a multiple-fluid Navier-Stokes model. *Coastal Engineering*, 57(9):779 – 794, 2010. ISSN 0378-3839.

[2] H. T. Ahn and M. Shashkov. Adaptive moment-of-fluid method. *Journal of Computational Physics*, 228:2792–2821, 2009.

[3] Hyung Taek Ahn and Mikhail Shashkov. Multi-material interface reconstruction on generalized polyhedral meshes. *J. Comput. Phys.*, 226(2): 2096–2132, 2007. ISSN 0021-9991.

[4] B. Ahrenholz, J. Tölke, and M. Krafczyk. Second-order accurate Lattice Boltzmann flow simulations in reconstructed porous media. *International Journal of Computational Fluid Dynamics*, 20 (6):369–377, 2006.

[5] Benjamin Ahrenholz. *Massively parallel simulations of multiphase- and multicomponent flows using lattice Boltzmann methods*. PhD thesis, TU Braunschweig, 2009.

[6] G.B. Airy. Tides and waves. *H.J. Rose, et al.. Encyclopaedia Metropolitana. Mixed Sciences*, 3, 1841.

[7] Eugenio Aulisa, Sandro Manservisi, and Ruben Scardovelli. A mixed markers and volume-of-fluid method for the reconstruction and advection of interfaces in two-phase and free-boundary flows. *J. Comput. Phys.*, 188(2):611–639, 2003. ISSN 0021-9991.

[8] Andrea Balzano. Evaluation of methods for numerical simulation of wetting and drying in shallow water flow models. *Coastal Engineering*, 34 (1-2):83 – 107, 1998.

[9] M. El Bettah, S.T. Grilli, M. Krafczyk, C.D.P Baxter, C. Janßen, and K. Bollinger. A Microfluidics Study of the Triggering of Underwater Landslides by Earthquakes. *submitted to IJOPE*, 2009.

[10] P. L. Bhatnagar, E. P. Gross, and M. Krook. A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems. *Phys. Rev.*, 94(3): 511–525, May 1954.

[11] B. Biausser, S.T. Grilli, P. Fraunie, and R. Marcer. Numerical analysis of the internal kinematics and dynamics of three-dimensional breaking waves on slopes. *International Journal of Offshore and Polar Engineering*, 14(4): 247–256, 2004.

[12] J. Boussinesq. Théorie des ondes et des remous qui se propagent le long d'un canal rectangulaire horizontal, en communiquant au liquide contenu dans ce canal des vitesses sensiblement pareilles de la surface au fond. *Journal de Mathématique Pures et Appliquées, Deuxième Série*, 17:55–108, 1872.

[13] M. Bouzidi, M. Firdaouss, and P. Lallemand. Momentum transfer of a lattice-Boltzmann fluid with boundaries. *Physics of Fluids*, 13:3452–3459, 2001.

[14] R. P. Brent. *Algorithms for Minimization Without Derivatives*. Prentice Hall, 1973. Reviewed in: American Scientist 61 (May-June 1973), 374; Mathematical Programming 4 (1973), 349; Computer J. 16 (1973), 314; Math. Comp. 28 (1974), 865-866.

[15] J. M. Buick and C. A. Greated. Gravity in a lattice Boltzmann model. *Phys. Rev. E*, 61(5): 5307–5320, May 2000.

[16] G. F. Carrier, T.T. Wu, and H. Yeh. Tsunami run-up and draw-down on a plane beach. *Journal of Fluid Mechanics*, 475:79–99, 2003.

[17] R. K. C. Chan and R. L. Street. A computer study of finite-amplitude water waves. *Journal of Computational Physics*, 6:68–94, 1970.

[18] S. Chapman and T. Cowling. *The mathematical theory of nonuniform gases: an account of the kinetic theory of viscosity, thermal conduction and diffusion in gases*. Cambridge University Press, 1990.

[19] Andrea Colagrossi and Maurizio Landrini. Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *Journal of Computational Physics*, 191(2):448–475, 2003. ISSN 0021-9991.

[20] Andrea Colagrossi, Matteo Antuono, and David Le Touzé. Theoretical considerations on the free-surface role in the smoothed-particle-hydrodynamics model. *Phys. Rev. E*, 79(5): 056701, May 2009.

[21] B. Crouse. *Lattice-Boltzmann Strömungssimulationen auf Baumdatenstrukturen*. PhD thesis, Lehrstuhl für Bauinformatik, Technische Universität München, 2003.

[22] B. Crouse, M. Krafcyzk, J. Tölke, and E. Rank. A LB-based approach for adaptive flow simulations. *International Journal of Modern Physics B*, 17:109 – 112, 2003.

[23] Dalrymple and Herault. Levee Breaching with GPU-SPHysics Code. In *Proceedings of the 4th international SPHERIC workshop Nantes, France, May, 27-29 2009*, 2009.

[24] R.A. Dalrymple and B.D. Rogers. Numerical modeling of water waves with the SPH method. *Coastal Engineering*, 53(2-3):141–147, 2006. ISSN 0378-3839. Coastal Hydrodynamics and Morphodynamics.

[25] B.J. Daly. Numerical study of two-fluid Rayleigh-Taylor instability. *Phys. Fluids*, 10: 297–307, 1967.

[26] Hendrik Dankowski. Lattice-Boltzmann slip boundary conditions for the modeling of transient flow around a ship hull, 2007. Student research project.

[27] Robert G. Dean and Robert A. Dalrymple. *Water wave mechanics for engineers and scientists*. World Scientific Publishing, Singapore, 1991. ISBN 981-02-0421-3.

[28] Lokenath Debnath. *Nonlinear Water Waves*. Academic Press, 1994.

[29] Dominique d'Humieres, Irina Ginzburg, Manfred Krafczyk, Pierre Lallemand, and Li-Shi Luo. Multiple Relaxation-Time Lattice Boltzmann models in three-dimensions. *Royal Society of London Philosophical Transactions Series A*, 360:437–451, 2002.

[30] Stefan Donath, Christian Feichtinger, Thomas Pohl, Jan Götz, and Ulrich Rüde. Localized Parallel Algorithm for Bubble Coalescence in Free Surface Lattice-Boltzmann Method. In *Euro-Par '09: Proceedings of the 15th International Euro-Par Conference on Parallel Processing*, pages 735–746, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-03868-6.

[31] V. Dyadechko and M. Shashkov. Moment-of-fluid interface reconstruction - revised. Technical Report Tech. Rep. LA-UR-07-1537, Los Alamos National Laboratory, 2007.

[32] Vadim Dyadechko and Mikhail Shashkov. Moment-of-fluid interface reconstruction. Technical report, Los Alamos National Laboratory, Oct 2005., 2005.

[33] Vadim Dyadechko and Mikhail Shashkov. Reconstruction of multi-material interfaces from moment data. *J. Comput. Phys.*, 227(11): 5361–5384, 2008. ISSN 0021-9991.

[34] Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A Hybrid Particle Level Set Method for Improved Interface Capturing. *J. Comput. Phys*, 183:83–116, 2002.

[35] Zhe Fan, Fenq Qiu, Arie Kaufman, and Suzanne Yoakum-Stover. GPU cluster for high performance computing. *In: Proceedings of ACM/IEEE Supercomputing Conference*, pages 47–59, 2004.

[36] A. Fayon and J. Happel. Effect of a cylindrical boundary on fixed rigid sphere in a moving fluids. *AIChE J*, 6(1):55–58, 1960.

[37] J. D. Fenton and M. M. Rienecker. A Fourier method for solving nonlinear water-wave problems: application to solitary wave interactions. *Journal of Fluid Mechanics*, 118:411–443, 1982.

[38] O. Filippova and D. Hänel. Boundary-Fitting and Local Grid Refinement for Lattice-BGK Models. *International Journal of Modern Physics C*, 9:1271–1279, 1998.

[39] Jannette B. Frandsen. Investigations of wave runup using a LBGK modeling approach. In *CMWR XVI proceedings of the XVI International Conference on Computational Methods in Water Resoures, Copenhagen, Denmark*, 2006.

[40] Jannette B. Frandsen. A simple LBE wave runup model. *Progess in Computational Fluid Dynamics*, 8:222–232, 2008.

[41] S. Freudiger and M. Krafczyk. VirtualFluids: A component based framework for parallel lattice Boltzmann simulations based on hierarchical block grids. Technical report, iRMB, TU Braunschweig, ICMMES, München, 2007.

[42] Sören Freudiger. *Entwicklung eines parallelen, adaptiven, komponentenbasierten Strömungskerns für hierarchische Gitter auf Basis des Lattice Boltzmann Verfahrens*. PhD thesis, TU Braunschweig, 2009.

[43] U. Frisch, D. d'Humières, B. Hasslacher, P. Lallemand, Y. Pomeau, and J.-P. Rivet. Lattice Gas Hydrodynamics in Two and Three Dimensions. *J. Complex Syst.*, 1:75–136, 1987.

[44] C. J. Galvin. Breaker Type Classification on three Laboratory Beaches. *J. of Geoph. Res.*, 73(12):3651–3659, 1968.

[45] D. Gaudlitz and N.A. Adams. The hybrid particle level-set method applied to two-phase flows. In *Proceedings of FEDSM2006, ASME Joint U.S.-European Fluids Engineering Summer Meeting*, 2006.

[46] Daniel Gaudlitz and Nikolaus A. Adams. On improving mass conservation properties of the hybrid particle level-set method. *Computers & Fluids*, 37:1320–1331, 2008.

[47] M. Geier, A. Greiner, and J. G. Korvink. Cascaded digital lattice Boltzmann automaton for high Reynolds number flow. *Phys. Rev. E*, 73: 066705, 2006.

[48] M. Geier, A. Greiner, and J. G. Korvink. Bubble functions for the lattice boltzmann method and their application to grid refinement. *The European Physical Journal Special Topics*, 171: 173–179, 2009.

[49] S. Geller. *Ein explizites Modell für die Fluid-Struktur-Interaktion basierend auf LBM und p-FEM*. PhD thesis, Fakultät Architektur, Bauingenieurwesen und Umweltwissenschaften der Technischen Universität Carolo-Wilhelmina zu Braunschweig, 2010.

[50] S. Geller, M. Krafczyk, J. Tölke, S. Turek, and J. Hron. Benchmark computations based on Lattice-Boltzmann, Finite Element and Finite volume Methods for laminar Flows. *Comp. and Fluids*, 35:888–897, 2006.

[51] Massimo Germano, Ugo Piomelli, Parviz Moin, and William H. Cabot. A dynamic subgrid-scale eddy viscosity model. *Phys. Fluids A*, 3 (7): 1760–1765, 1991.

[52] M. Geveler, D. Ribbrock, D. Göddeke, and S. Turek. Lattice–Boltzmann simulation of the shallow–water equations with fluid–structure interaction on multi– and manycore processors. In R. Keller, D. Kramer, and J. Weiss, editors, *Facing the Multicore–Challenge*, Akademiekonferenzen, pages 99–112. Heidelberg University Press, March 2010.

[53] I. Ginzburg and D. D'Humieres. Multireflection boundary conditions for lattice Boltzmann models. *Physical Review E*, 68(6):066614.1–066614.30, December 2003.

[54] I. Ginzburg and G. Wittum. Two-Phase Flows on Interface Refined Grids Modeled with VOF, Staggered Finite Volumes, and Spline Interpolants. *J. Comput. Phys.*, 166(2):302–335, 2001. ISSN 0021-9991.

[55] Irina Ginzburg and Konrad Steiner. Lattice Boltzmann model for free-surface flow and its application to filling process in casting. *J. Comput. Phys.*, 185(1):61–99, 2003. ISSN 0021-9991.

[56] Irina Ginzburg, Frederik Verhaeghe, and Dominique d'Humières. Two-Relaxation-Time Lattice Boltzmann Scheme: About Parametrization,Velocity, Pressure and Mixed Boundary Conditions. *Communications in computational physics*, 3:427–478, 2008.

[57] J. Glimm, O. McBryan, R. Menikoff, and D.H. Sharp. Front tracking applied to Rayleigh-Taylor instability. *SIAM J. Sci. Stat. Comput.*, 7:230–251, 1986.

[58] James Glimm, John W. Grove, Xiao Lin Li, Keh-Ming Shyue, Yanni Zeng, and Qiang Zhang. Three Dimensional Front Tracking. *SIAM J. Sci. Comp*, 19:703–727, 1995.

[59] R. Glowinski, P. Le Tallec, M. Ravachol, and V. Tsikkinis. Numerical solution of the Navier-Stokes equations modelling the flow of two incompressible nonmiscible viscous fluids. *Finite Elements in Fluids*, 8:137–163, 1992.

[60] Hitoshi Gotoh, Hiroyuki Ikari, Tetsu Memita, and Tetsuo Sakai. Lagrangian particle method for simulation of wave overtopping on a vertical seawall. *Coastal Engineering Journal*, 47: 157–181, 2005.

[61] N. Grenier, M. Antuono, A. Colagrossi, D. Le Touzé, and B. Alessandrini. An Hamiltonian interface SPH formulation for multi-fluid and free surface flows. *J. Comput. Phys.*, 228 (22):8380–8393, 2009. ISSN 0021-9991.

[62] S. Grilli, P. Guyenne, and F. Dias. A fully nonlinear model for three-dimensional overturning waves over arbitrary bottom. *International Journal for Numerical Methods in Fluids*, 35(7):829 – 867, 2001.

[63] S. T. Grilli and R. Subramanya. Numerical Modeling of Wave Breaking Induced by Fixed or Moving Boundaries. *Computational Mechanics*, 17(6):374–391, 1996.

[64] S.T. Grilli and J. Horrillo. Numerical Generation and Absorption of Fully Nonlinear Periodic Waves. *Journal of Engineering Mechanics*, 123(10):1060–1069, 1997.

[65] S.T. Grilli, I.A. Svendsen, and R. Subramanya. Breaking Criterion and Characteristics for Solitary Waves on Slopes. *Journal of Waterway Port Coastal and Ocean Engineering*, 123(3): 102–112, 1997.

[66] Denis Gueyffier, Jie Li, Ali Nadim, Ruben Scardovelli, and Stéphane Zaleski. Volume-of-fluid interface tracking with smoothed surface stress methods for three-dimensional flows. *Journal of Computational Physics*, 152(2):423–456, 1999. ISSN 0021-9991.

[67] S. Guignard, S. T. Grilli, R. Marcer, and V. Rey. Computation of shoaling and breaking waves in nearshore areas by the coupling of BEM and VOF methods. *Proc. 9th Offshore and Polar Engng. Conf.*, III:304–309, 1999.

[68] Andrew K. Gunstensen, Daniel H. Rothman, Stéphane Zaleski, and Gianluigi Zanetti. Lattice Boltzmann model of immiscible fluids. *Phys. Rev. A*, 43(8):4320–4327, Apr 1991.

[69] Z. Guo, C. Zheng, and B. Shi. Discrete lattice effects on the forcing term in the lattice Boltzmann method. *Physical Review E*, 65(4): 046308.1 – 046308.6, April 2002.

[70] Johannes Habich, Thomas Zeiser, G. Hager, and G. Wellein. Performance analysis and optimization strategies for a D3Q19 Lattice Boltzmann Kernel on nVIDIA GPUs using CUDA. *Special issue of "Advances in Engineering Software and Computers & Structures"*, 2010.

[71] Takahiro Harada and Yoichiro Kawaguchi. Smoothed Particle Hydrodynamics on GPUs. *Proceedings of Computer Graphics International*, 2007.

[72] J. Hardy, O. de Pazzis, and Y. Pomeau. Molecular dynamics of a latice gas: transport properties and time correlation functions. *Physical Review A*, 13:1949–1961, 1976.

[73] F. H. Harlow and J.E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids*, 8: 2182–2189, 1965.

[74] J. Harris and S.T. Grilli. Coupling of NWT and large eddy simulation for wave-induced sediment transport. *Proc. 20th Offshore and Polar Engng. Conf.*, 2010.

[75] Ami Harten. High Resolution Schemes for Hyperbolic Conservation Laws. *Journal of Computational Physics*, 49:357–393, 1983.

[76] M. J. Harvey, G. De Fabritiis, and G. Giupponi. Accuracy of the lattice-Boltzmann method using the Cell processor. *Phys. Rev. E*, 78(5): 056702, Nov 2008.

[77] Dalton J. E. Harvie and David F. Fletcher. A New Volume of Fluid Advection Algorithm: The Stream Scheme. *Journal of Computational Physics*, 162:1–32, 2000.

[78] X. He and L.-S. Luo. Lattice Boltzmann model for the incompressible Navier-Stokes equation. *Journal of Statistical Physics*, 88: 927–944, 1997.

[79] A. Herault, G. Bilotta, and R. A. Dalrymple. SPH on GPU with CUDA. *Journal of Hydraulic Research*, 48(Extra Issue):74–79, 2010. ISSN 0022-1686.

[80] F. Higuera and J. Jiménez. Boltzmann approach to lattice gas simulations. *Europhysics Letters*, 9:663–668, 1989.

[81] F. J. Higuera, S. Succi, and R. Benzi. Lattice Gas Dynamics with Enhanced Collisions. *Europhysics Letters*, 9:345–349, 1989.

[82] Hiromaru Hirakuchi, Ryoichi Kajima, and Takashi Kawaguchi. Application of a Piston-Type Absorbing Wavemaker to Irregular Wave Experiments. *Coastal Engineering in Japan*, 33 (1):11–24, 1990.

[83] C.W. Hirt and B.D. Nichols. Volume of fluid method for dynamics of free boundaries. *Journal of Computational Physics*, 39:201–221, 1981.

[84] J. Hron and S. Turek. A monolithic FEM solver for ALE formulation of fluid structure interaction with configurations for numerical benchmarking. In *Proceedings of the International Conference on Computational Methods for Coupled Problems in Science and Engineering*, Santorini, 2005.

[85] B. Hübner, E. Walhorn, and D. Dinkler. A monolithic approach to fluid-structure interaction using space-time finite elements. *Computer Methods in Applied Mechanics and Engineering*, 193:2087–2104, 2004.

[86] K. Iglberger, N. Thürey, and U. Rüde. Simulation of moving particles in 3d with the lattice boltzmann method. *Computers & Mathematics with Applications*, 55(7):1461–1468, 2008.

[87] Salvador Izquierdo and Norberto Fueyo. Characteristic nonreflecting boundary conditions for open boundaries in lattice Boltzmann methods. *Phys. Rev. E*, 78(4):046707, Oct 2008.

[88] C. Janßen. Free surface tracking with the Lattice Boltzmann Method. Student research project, 2007.

[89] C. Janßen and S. Grilli. Modeling of Wave Breaking and Wave-Structure Interactions by Coupling of Fully Nonlinear Potential Flow and Lattice-Boltzmann Models. *submitted to: ISOPE Proceedings*, 2010.

[90] C. Janßen and M. Krafczyk. A lattice Boltzmann approach for free-surface-flow simulations on non-uniform block-structured grids. *Computers and Mathematics with Applications*, 2009.

[91] M. Junk, A. Klar, and L.-S. Luo. Asymptotic analysis of the lattice Boltzmann equations. *Journal of Computational Physics*, 210 (2):676–704, 2005.

[92] Micky Kelager. Lagrangian Fluid Dynamics Using Smoothed Particle Hydrodynamics, 2006.

[93] David Kirk and Wen-Mei W. Hwu. *Programming Massively Parallel Processors: A Hands-On Approach*. Morgan Kaufmann Publishers Inc, 2010.

[94] K.M.T. Kleefsman. *Water impact loading on offshore structures - a numerical study*. PhD thesis, University of Groningen, 2005.

[95] C. Körner, M. Thies, T. Hofmann, N. Thürey, and U. Rüde. Lattice Boltzmann Model for Free Surface Flow for Modeling Foaming. *Journal of Statistical Physics*, 121(18):179–196, October 2005.

[96] D. J. Korteweg and G. de Vries. On the Change of Form of Long Waves Advancing in a Rectangular Canal, and on a New Type of Long Stationary Waves. *Philosophical Magazine*, 39: 422–443, 1895.

[97] M. Krafczyk. Gitter-Boltzmann-Methoden: Von der Theorie zur Anwendung, 2001. Habilitation Thesis.

[98] M. Krafczyk, J. Tölke, and L.-S. Luo. Large-eddy simulations with a multiple-relaxation-time LBE model. *Int. J. Mod. Phys. B*, 17: 33–39, 2003.

[99] Andreas Kölke. *Modellierung und Diskretisierung bewegter Diskontinuitäten in randgekoppelten Mehrfeldsystemen*. PhD thesis, Technische Universität Braunschweig, 2005.

[100] Christophe Lachaume, Benjamin Biausser, Stéphan T. Grilli, and Stéphan Guignard. Modeling of breaking and post-breaking waves on slopes by coupling of BEM and VOF methods. In *In Proc. 13th Intl Offshore and Polar Engng Conf*, pages 353–359, 2003.

[101] Anthony J.C. Ladd. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation. *Journal of Fluid Mechanics*, 271:285–309, 1994.

[102] Pierre Lallemand and Li-Shi. Luo. Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability. *Physical Review*, E 61:6546–6562, June 2000.

[103] Pierre Lallemand, Li-Shi Luo, and Yan Peng. A lattice Boltzmann front-tracking method for interface dynamics with surface tension in two dimensions. *Journal of Computational Physics*, 226(2):1367–1384, 2007. ISSN 0021-9991.

[104] R. P. Leonard. *Universal limiter for transient interpolation modeling of the advective transport equations: the ULTIMATE conservative difference schemes*. National Aeronautics and Space Administration, 1988.

[105] R. J. LeVeque. High-resolution conservative algorithms for advection in incompressible flow. *SIAM J. Numer. Anal.*, 33:627–665, 1996.

[106] D. K. Lilly. A proposed modification of the Germano subgrid-scale closure methods. *Phys. Fluids A 4*, 4:633–635, 1991.

[107] Jan Linxweiler, Manfred Krafczyk, and Jonas Tölke. Highly interactive computational steering for coupled 3D flow problems utilizing multiple GPUs. *Computing & Visualization in Science*, 2011 accepted for publication.

[108] M. S. Longuet-Higgins. Mass transport in water waves. *Philosphical Transactions of the Royal Society of London*, A:535–581, 1953.

[109] Patrick J. Lynett, Tso-Ren Wu, and Philip L. F. Liu. Modeling wave runup with depth-integrated equations. *Coastal Engineering*, 46 (2):89 – 107, 2002.

[110] E. Marchi. On the free overfall. *Journal of Hydraulic Research*, 31:777–790, 1993.

[111] S. Marrone, A. Colagrossi, D. Le Touzé, and G. Graziani. Fast free-surface detection and level-set function definition in SPH solvers. *J. Comput. Phys.*, 229(10):3652–3663, 2010. ISSN 0021-9991.

[112] J. C. Martin and W. J. Moyce. Part IV. An Experimental Study of the Collapse of Liquid Columns on a Rigid Horizontal Plane. *Royal Society of London Philosophical Transactions Series A*, 244:312–324, March 1952.

[113] G. McNamara and G. Zanetti. Use of the Boltzmann equation to simulate lattice-gas automata. *Physical Review Letters*, 61:2332–2335, 1988.

[114] Renwei Mei, Li-Shi Luo, Pierre Lallemand, and Dominique d'Humieres. Consistent initial conditions for lattice Boltzmann simulations. *Computers & Fluids*, 35(8-9):855 – 862, 2006. ISSN 0045-7930.

[115] Markus Meier, George Yadigaroglu, and Brian L. Smith. A novel technique for including surface tension in PLIC-VOF methods. *European Journal of Mechanics - B/Fluids*, 21(1):61 – 73, 2002. ISSN 0997-7546.

[116] G.H. Miller and P. Colella. A conservative three-dimensional eulerian method for coupled solid-fluid shock capturing. *Journal of Computational Physics*, 183:26–82, 2002.

[117] L. M. Milne-Thomson. *Theoretical hydrodynamics*. Macmillan, New York, NY, 1960.

[118] J. J. Monaghan. Simulating free surface flows with SPH. *J. Comput. Phys.*, 110(2):399–406, 1994. ISSN 0021-9991.

[119] J. J. Monaghan and A. Kos. Scott Russell's wave generator. *Physics of Fluids*, 12:622–630, March 2000.

[120] G. Mosqueira, L. Cueto-Felgueroso, I. Colominas, F. Navarrina, and M. Casteleiro. SPH approaches for free surface flows in engineering applications. In *In: Proceedings of WCCM V, Fifth World Congress on Computational Mechanics*, 2002.

[121] T. Ménard, S. Tanguy, and A. Berlemont. Coupling level set/VOF/ghost fluid methods: Validation and application to 3D simulation of the primary break-up of a liquid jet. *International Journal of Multiphase Flow*, 33(5):510 – 524, 2007.

[122] T. Nakayama and M. Mori. An Eulerian finite element method for timedependent free surface problems in hydrodynamics. *Int. J. Num. Meth. Fluids*, 22:175–194, 1996.

[123] N.Q. Nguyen and A.J.C. Ladd. Sedimentation of hard-sphere suspensions at low Reynolds number. *J. Fluid Mech*, 535:73 – 104, 2004.

[124] F. Nicoud and F. Ducros. Subgrid-scale stress modelling based on the square of the velocity gradient tensor. *Flow, Turbulence and Combustion*, 62:183–200, 1999. ISSN 1386-6184. URL `http://dx.doi.org/10.1023/A:1009995426001`. 10.1023/A:1009995426001.

[125] W. F. Noh and P. Woodward. SLIC (simple line interface calculation). In A. I. van de Vooren and P. J. Zandbergen, editors, *Proceedings of 5th International Conference of Fluid Dynamics*, volume 59 of *Lecture Notes in Physics*, pages 330–340. Springer, 1976.

[126] nVIDIA. nVIDIA CUDA. URL `http://www.nvidia.com/object/cuda_home_new.html`.

[127] *NVIDIA CUDA Programming Guide*. nVIDIA, `http://www.nvidia.com/object/cuda_develop.html`, 2010.

[128] Stanley Osher and James A. Sethian. Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations. *Journal of Computational Physics*, 79: 12–49, 1998.

[129] B.J. Parker and D.L. Youngs. Two and three dimensional Eulerian simulation of fluid flow with material interfaces. Technical report, UK Atomic Weapons Establishment, February 1992.

[130] J. E. Pilliod. An analysis of piecewise linear interface reconstruction algorithms for volume-of-fluid methods. Master's thesis, University of California, Davis, 1992.

[131] James Edward Pilliod, Jr. and Elbridge Gerry Puckett. Second-order accurate volume-of-fluid algorithms for tracking material interfaces. *J. Comput. Phys.*, 199(2):465–502, 2004. ISSN 0021-9991.

[132] E. G. Puckett and J. S. Saltzman. A 3D adaptive mesh refinement algorithm for multimaterial gas dynamics. *Phys. D*, 60(1-4):84–93, 1992.

[133] E.G. Puckett. A volume of fluid interface tracking algorithm with applications to computing shock wave rarefraction. In *Proceedings of the 4th International Symposium on Computational Fluid Dynamics*, pages 933 – 938, 1991.

[134] Y. H. Quian, D. d'Humieres, and P. Lallemand. Lattice BGK models for Navier Stokes equations. *Europhysics Letters*, 17:479–484, 1992.

[135] M. Rheinländer. A Consistent Grid Coupling Method for Lattice-Boltzmann Schemes. *Journal of Statistical Physics*, 121(1-2):49–74, 2005.

[136] William J. Rider and Douglas B. Kothe. Reconstructing volume tracking. *Journal of Computational Physics*, 141:141–112, 1998.

[137] V. Roubtsova and R. Kahawita. The SPH technique applied to free surface flows. *Computers & Fluids*, 35(10):1359–1371, 2006. ISSN 0045-7930.

[138] Murray Rudman. Volume-tracking methods for interfacial flow calculations. *International Journal for Numerical Methods in Fluids*, 24:671–691, 1997.

[139] A. Salih and S. Ghosh Moulic. A level set formulation for the numerical simulation of impact of surge fronts. *SADHANA - Academy Proceedings in Engineering Sciences*, 31:697–707, 2006.

[140] Jason Sanders and Edward Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley Professional, 2010.

[141] J. Sauer. *Instationär kavitierende Strömungen - Ein neues Modell, basierend auf Front Capturing (VOF) und Blasendynamik*. PhD thesis, Universität Karlsruhe, 2000.

[142] Marco Schauer. Fluid Struktur Interaktion auf Basis der Lattice Boltzmann und Finite Elemente Methode in 3D, 2007. Student research project.

[143] M. Schelkle. *LB-Verfahren zur Simulation dreidimensionaler Zweiphasen-Strömungen mit freien Oberflächen*. PhD thesis, Universität Stuttgart, 1996.

[144] L. Schiller and A.Z. Naumann. Ueber die grundlegenden Berechnungen bei der Schwerkraftaufbereitung. *Zeitschrift des Vereines Deutscher Ingenieure*, 77(12):318–320, 1933.

[145] Martin Schönherr, Kostyantyn Kucher, Martin Geier, Maik Stiebler, Sören Freudiger, and Manfred Krafczyk. Multi-thread implementations of the lattice Boltzmann method on non-uniform grids for CPUs and GPUs. *Computers and Mathematics with Applications*, 61(12), June 2011.

[146] Xiaowen Shan, Xue-Feng Yuan, and Hudong Chen. Kinetic theory representation of hydrodynamics: a way beyond the Navier-Stokes equation. *J. Fluid Mech.*, 550:413–441, 2006.

[147] Nathanael Shärli, Stéphane Ducasse, Oscar Nierstrasz, and Andrew Black. Traits: Composable units of behavior. In *Cardelli, L. (ed.) ECOOP 2003. LNCS*, volume 2743, pages 248–274. Springer, 2003.

[148] W. Shyy, H.S. Udaykumar, M.M. Rao, and R.W. Smith. *Computational Fluid Dynamics with Moving Boundaries*. Taylor&Francis, 1996.

[149] Joseph Smagorinsky. General Circulation experiments with the primitive equations I. The basic experiment. *Monthly Weather Review*, 91:99–164, 1963.

[150] SPH. SPHysics - SPH Free-surface Flow Solver (Open-Source Smoothed Particle Hydrodynamics code). URL `http://wiki.manchester.ac.uk/sphysics/`.

[151] Maik Stiebler. Fruitful discussions on the perturbation approach and MRT collision operators, iRMB, TU Braunschweig, July 2010.

[152] Sauro Succi. *The Lattice Boltzmann Equation*. Clarendon Press Oxford, 2001.

[153] S. Tanaka and K. Kashiyama. ALE finite element method for FSI problems with free surface using mesh re–generation method based on background mesh. *International Journal of Computational Fluid Dynamics*, 20:229–236, 2006.

[154] Ven te Chow. *Open-Channel Hydraulics*. MC Graw-Hill, 1959.

[155] Nils Thürey and Ulrich Rüde. Free Surface Lattice-Boltzmann fluid simulations with and without level sets. *Proc. of Vision, Modelling, and Visualization VMV*, pages 199–207, 2004.

[156] Nils Thürey and Ulrich Rüde. Stable free surface flows with the lattice Boltzmann method on adaptively coarsened grids. *Computing and Visualization in Science*, page ., 2008.

[157] Guido Thömmes, Mohammed Seaid, and Mapundi K. Banda. Lattice boltzmann methods for shallow water flow applications. *Int. J. Numer. Meth. Fluids*, 55:673–692, 2007.

[158] Jonas Tölke and Manfred Krafczyk. Implementation of a Lattice Boltzmann kernel using the Compute Unified Device Architecture developed by nVIDIA. *Computing and Visualization in Science*, 1:29–39, 2008.

[159] Jonas Tölke and Manfred Krafczyk. TeraFLOP computing on a desktop PC with GPUs for 3D CFD. *International Journal of Computational Fluid Dynamics*, 22:443–456, 2008.

[160] Kevin Tubbs. *Lattice Boltzmann Modeling for shallow water equations using high performance computing*. PhD thesis, Lousiana State University, 2010.

[161] J. Tölke. *Gitter-Boltzmann-Verfahren zur Simulation von Zweiphasenströmungen*. PhD thesis, TU München, München, 2001.

[162] O. Ubbink and R.I.Issa. A method for capturing sharp fluid interfaces on arbitrary meshes. *Journal of Computational Physics*, 153:26–50, 1999.

[163] S. O. Unverdi and G. Tryggvason. A front-tracking methd for viscous incompressible multi-fluid flows. *Journal of Computational Physics*, 100:25–37, 1992.

[164] Sonja Uphoff. Validation of the LES approach Wale in a lattice Boltzmann framework on the Ahmed body benchmark. Oral presentation, ICMMES 2009, 2009.

[165] B. van Leer. Towards the Ultimate Conservative Difference Scheme, V. A Second Order Sequel to Godunov's Method. *Journal of Computational Physics*, 32:101–136, 1979.

[166] A.E.P. Veldman. ComFLOW - dambreak experiment, 2005. URL `http://www.math.rug.nl/~veldman/comflow/dambreak.html`.

[167] Tomasz Waclawczyk and Tadeusz Koronowiczy. Modeling of the wave breaking with CICSAM and HRIC high-resolution schemes. In *European Conference on Computational Fluid Dynamics, ECCOMAS CFD 2006*, 2006.

[168] Tomasz Waclawczyk and Tadeusz Koronowiczy. Comparison of CICSAM and HRIC high-resolution schemes for interface capturing. *Journal of Theoretical and Applied Mechanics*, 46(2):325–345, 2008.

[169] Elmar Walhorn. *Ein simultanes Berechnungsverfahren für Fluid-Struktur-Wechselwirkungen mit finiten Raum-Zeit-Elementen*. PhD thesis, Institut für Statik, TU Braunschweig, 2002.

[170] Xiaoming Wei, Ye Zhao, Zhe Fan, Wei Li, Feng Qiu, Suzanne Yoakum-Stover, and Arie Kaufman. Lattice-based flow field modeling. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):719–729, 2004.

[171] M. Weickert, G. Teike, O. Schmidt, and M. Sommerfeld. Investigation of the LES WALE turbulence model within the lattice boltzmann framework. *Computers and Mathematics with Applications*, 59:2200–2214, 2009.

[172] J.E. Welch, F.W. Harlow, J.P. SHannon, and B.J. Daly. The MAC method: A computing technique for solving viscous, incompressible, transient fluid flow problems involving free surfaces. *Los Alamos Scientific Laboratory Report*, LA-3425, 1966.

[173] Rik Wemmenhove. *Numerical Simulation of Two-Phase Flow in Offshore Environments*. PhD thesis, Rijksuniversiteit Groningen, 2008.

[174] Rik Wemmenhove, Rune Gladso, Bogdan Iwanoswki, and Marc Lefranc. Comparison of CFD calculations and experiment for the dambreak experiment with one flexible wall. In *Proceedings of the Twentieth (2010) International Offshore and Polar Engineering Conference*, 2010.

[175] Jan Wienke. *Druckschlagbelastung auf schlanke zylindrische Bauwerke durch brechende Wellen*. PhD thesis, TU Braunschweig, 2001.

[176] Dieter A. Wolf-Gladrow. *Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction*. Springer Verlag, 2000.

[177] S.C. Yim, H. Yeh, and D. Cox. International collaborative tsunami, storm surge, and wave structure interaction research opportunities using the oregon state multidirectional wave basin and large wave flume. *Proceedings of The 36th Joint Panel Meeting on Wind and Seismic Effects*, pages 1–11, 2004.

[178] D.L. Youngs. Time-dependent multimaterial flow with large fluid distortion. In K. Morton and M. Baines, editors, *Numerical Methods for Fluid Dynamics*, pages 273–285. Academic Press, 1982.

[179] D. Yu, R. Mei, L.S. Luo, and W. Shyy. Viscous flow computations with the method of lattice Boltzmann equation. *Progess In Aerospace Sciences*, 39:329–367, 2003.

[180] S.T. Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of Computational Physics*, 31:335–362, 1979.

[181] Q. Zhang and P. L.-F. Liu. A new interface tracking method: The polygonal area mapping method. *Journal of Computational Physics*, 227:4063–4088, 2008.

[182] Yanci Zhang, Barbara Solenthaler, and Renato Pajarola. GPU accelerated SPH particle simulation and rendering. In *SIGGRAPH '07: ACM SIGGRAPH 2007 posters*, New York, 2007. ACM.

[183] Y. Zhao and et al. Melting and flowing in multiphase environments. *Computers & Graphics*, 30(4):519–528, 2006.

[184] J. Zhou. *Lattice Boltzmann Methods for Shallow Water Flows*. Springer, 2003. ISBN 3540407464.

[185] Hongbin Zhu, Xuehui Liu, Youquan Lui, and Enhua Wu. Simulation of miscible binary mixtures based on lattice Boltzmann method. *Computer Animation and Virtual Worlds*, 17:403–410, 2006.

Nomenclature

$\mathbf{e}_i$ ..... discrete particle velocity

$\mathbf{v}$ ..... Velocity vector

$\mathbf{v}^I, p^I$ ..... Inviscid velocity and pressure fields

$\mathbf{v}^P, p^P$ ..... Viscous velocity and pressure fields

$f(t, \mathbf{x}, \boldsymbol{\xi})$ ..... particle distribution function

$f_i(t, \mathbf{x})$ ..... discrete particle distribution function

$f_i^{eq}(\rho, \mathbf{v})$ ..... Maxwellian equilibrium distribution function

v ..... Magnitude of the velocity vector $\mathbf{u}$

AdHoC ..... Adaptive hierachical object code with C

CFD ..... Computational Fluid Dynamics

CICSAM ..... Compressive Interface Capturing Scheme for Arbitrary Meshes

CUDA ..... C Unified Device Architecture

DFSBC ..... Dynamic free surface boundary condition

FHP ..... Frisch, Hasslacher, Pomeau lattice gas automaton

FNPF ..... Fully Nonlinear Potential Flow

FSI ..... Fluid-Structure Interaction

GPU ..... Graphics Processing Unit

HPP ..... Hardy, Pomeau, de Pazzis lattice gas automaton

KdV ..... Korteweg-de-Vries

KFSBC ..... Kinematic free surface boundary conditions

LBM ..... Lattice Boltzmann Method

LGA ..... Lattice gas automaton

LS ..... Level Set

MAC ..... Marker and cell

MOF ..... Moment of Fluid

MRT ..... Multiple Relaxation Time

MUSCL ..... Monotone Upstream-centered Schemes for Conservation Laws

NWT ..... Numerical Wave Tank

PDF ..... particle distribution function

PE ..... Physics Engine

PLIC ..... Piecewise Linear Interface Calculation

SLIC ..... Simple Linear Interface Calculation

SPH ..... Smoothed particle hydrodynamics

SRT ..... Single Relaxation Time

SWE ..... Shallow water equation

TRT ..... Two Relaxation Time

VOF ..... Volume of Fluid

WaLBerla ..... Widely Applicable Lattice Boltzmann Solver from Erlangen

Index

List of Figures

List of Tables

List of test cases

List of Algorithms

The end.